Army Research Laboratory

Discrete Optimization in Chemical Space Reference Manual

by B. C. Rinderspacher

ARL-TR-6202 October 2012

NOTICES

Disclaimers

Army Research Laboratory
Aberdeen Proving Ground, MD 21005

ARL-TR-6202 **October 2012**

Discrete Optimization in Chemical Space Reference Manual

B. C. Rinderspacher
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE						Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**						
1. REPORT DATE *(DD-MM-YYYY)* October 2012	**2. REPORT TYPE** Interim					**3. DATES COVERED (From - To)** 2009–2011
4. TITLE AND SUBTITLE Discrete Optimization in Chemical Space Reference Manual						**5a. CONTRACT NUMBER**
						5b. GRANT NUMBER
						5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) B. C. Rinderspacher						**5d. PROJECT NUMBER**
						5e. TASK NUMBER
						5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: RDRL-WMM-G Aberdeen Proving Ground, MD 21005						**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-6202
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)						**10. SPONSOR/MONITOR'S ACRONYM(S)**
						11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.						
13. SUPPLEMENTARY NOTES						
14. ABSTRACT We present the manual for using the constrained inverse molecular design program. The program uses an object-oriented approach implemented in C++. The manual includes instructions on setting up constrained optimizations of substitutional frameworks and the full application programming interface (API) necessary for extending and developing new capabilities.						
15. SUBJECT TERMS Molecular optimization, software documentation						
16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**		**19a. NAME OF RESPONSIBLE PERSON** B. C. Rinderspacher
a. REPORT UNCLASSIFIED	**b. ABSTRACT** UNCLASSIFIED	**c. THIS PAGE** UNCLASSIFIED	UU	413		**19b. TELEPHONE NUMBER** *(Include area code)* (410) 306-2811

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

1. Introduction

1.1 Usage

These modules include a reference counted Linear Algebra namespace (linear_algebra) for easy manipulation and a general substitution class based on recursion (ChemIdent). In the main driver, the function calc_property() executes *./property_script*. The secondary driver executes an energy computation via *./energy_script*.

To test the program, copy all files from the test directory to your working directory and invoke

```
/path_to_src/discrete_opt ftc-input.inp
```

The result should be a simple optimization of the hyperpolarizibility.

calc_property() will generate a *.zmat* file which will hold the Z-matrix of the current molecule. The prefix will be the current number of visited configurations followed by the current conformation and overall structures visited. See calc_property() for details. *energy_script* must return *.rconsts, .rvars, .energy* files on a successful run.

- *.rconsts* must contain the optimized constants of the Z-matrix from the geometry optimization.
- *.rvars* must contain the optimized variables of the Z-matrix from the geometry optimization.
- *.result* holds the result of the objective function.
- *.energy* holds the energy of the final geometry.

property_script must return *.result* and potentially *.penalty* files on a successful run.

- *.result* holds the result of the objective function.
- *.penalty* contains a penalty value.

To access the program flow go to main(). For information on the representation of molecules see class ChemIdent. For information on the input file syntax see ChemIdent::ChemIdent(stringstream& s).

1.1.1 Command Line Options

Parameters:

$-p$ Switches pruning or heuristics on.

See also:

noprune, simple_prune, reorder_general_base.

Parameters:

-sub-method <method>
-sm <method> details the sub-method to be used. Currently this includes *binary_line_search, GBLS (gen_base_LS), GBGLS (gen_base_grad_LS), SD, steepest_descent (binary_steepest_descent)*
-m <method> Supplies the method to be used. Options include all listed under sub-methods and *GBEN (gen_base_entropic), GDMC (binary_gdmc)*
-T <temperature> provides a fictitious temperature for GDMC calculations.
-TS <steps> provides the number of steps to tighten selection criteria for GDMC.
-MS <steps> Maximum number of evaluations used in GBEN and GDMC.

1.2 Installation

- Untar and unzip the tarball. The source will be extracted into `discrete_opt/`.
- Change direcory to discrete_opt/src and edit Doxyfile to change the documentation path to your liking.
- To build the code, execute: `make`
- For Documentation execute:

```
make docs
```

- For the debugable version execute:

```
make discrete_opt.d
```

MakeVars and Makefile contain the compilation information which you may want to change. The executable will be named *discrete_opt*.

2. Discrete Optimization in Chemical Space Module Documentation

2.1 Discrete Optimization of Chemical Space

Files

- file Library_data.cc
  Implementations of the Library class for discrete spaces.

Classes

- class has_gradients_data < X >
  Abstract class for discrete optimizations.
- class has_hessians_data < X >
  Abstract class for discrete optimizations.
- class Library_data
  Abstract class for discrete optimizations.

Functions

- class Library_data & Library_data::operator= (const Library_data &d)
  Assignment operator.
- void Library_data::set_Name (const string &A) const
  Sets the name as written for the purpose of writing files or other I/O.

2.1.1 Function Documentation

2.1.1.1 Library_data & Library_data::operator= (const Library_data & *d*)
[inherited]

2.1.1.2 void Library_data::set_Name (const string & *A*) const
[inherited]

2.2 Linear Algebra Package with Sparse Implementations

Files

- file ABmBA.hh
- file ABpBA.hh
- file asymmetric.decl
  Provide declarations for antisymmetric full matrices.
- file asymmetric.h
  Provide manipulations for antisymmetric full matrices.
- file class_extensions.decl
  Declarations of general expansions for LA.
- file class_extensions.hh
  General expansions for LA.
- file linear_algebra.decl
  Namespace Class Declarations.
- file mat_asym_full.decl
  Provide declarations for antisymmetric full matrices.
- file mat_asym_full.h
  Provide manipulations for antisymmetric full matrices.
- file mat_sym_sparse.decl
  Provide declarations for antisymmetric sparse matrices.
- file mat_sym_sparse.h
  Provide functionality for symmetric sparse matrices.
- file mat_full.decl
  Provide declarations for full matrix and vector interactions.
- file mat_sparse.decl
  Provide declarations for sparse matrix and vector interactions.
- file mat_sparse.h
  Provide functionality for sparse matrix and vector interactions.
- file mat_sym_full.decl
  Provide declarations for symmetric full matrices.

- file mat_sym_full.h

  Provide manipulatins for symmetric full matrices.

- file mat_sym_sparse_eigen.decl

  Eigenvalue related declarations.

- file mat_sym_sparse_eigen.h

  Eigenvalue related routines.

- file matrix.decl

  Declares fundamentals of matrices.

- file matrix.h

  Describes fundamentals of matrices.

- file polynomial.decl

  Class declarations for polynomial.

- file polynomial.h

  Implementations for a class of polynomials.

- file refcount.decl

  Contains the declarations to a reference counted vector class as well as general reference counting.

- file refcount.h

  Contains a reference counted vector class as well as general reference counting.

- file sparse_vector.decl

  Provide declarations for sparse vector interactions.

- file sparse_vector.h

  Provide functionality for sparse vectors.

- file sparse_vector_infty.decl

  Provide declarations for sparse vector interactions of infinite dimensions.

- file sparse_vector_infty.h

  Provide functionality for sparse vectors of infinite dimension.

- file square.decl

  Declares fundamentals of square matrices.

- file square.h

  Describes fundamentals of matrices.

- file symmetric.decl
  Declares fundamentals of matrices.

- file symmetric.h
  Describes fundamentals of symmetric matrices.

- file trace_AB.hh
  get the trace of matrix products.

Namespaces

- namespace linear_algebra
  Linear algebra Namespace.

Functions

- void linear_algebra::mat_sym_sparse::gen_mat (const sparse_vector< TN > &a, const sparse_vector< TN > &b, TN correction)
  Produce the necessary $ij^* + ji^*$.

- mat_sparse< TN > linear_algebra::mat_sym_sparse::householder (vector< TN > &diagonal, vector< TN > &subdiagonal)
  This routine produces the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.

- mat_sparse< TN > linear_algebra::mat_sym_sparse::householder (refvector< TN > &diagonal, refvector< TN > &subdiagonal)
  This routine produces the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.

- void gen_mat (const sparse_vector< TN > &a, const sparse_vector< TN > &b, TN correction)
  Produce the necessary $ij^* + ji^*$.

- mat_sparse< TN > householder (vector< TN > &diagonal, vector< TN > &subdiagonal)
  This routine produces the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.

- mat_sparse< TN > householder (refvector< TN > &diagonal, refvector< TN > &subdiagonal)
  This routine produces the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.

- template<class TN> void std::operator>> (const vector< TN > &a, ostream &OUT)
- template<typename TN> void std::operator>> (const TN &a, ostream &OUT)

2.2.1 Function Documentation

2.2.1.1 **void gen_mat (const sparse_vector< TN > & *a*, const sparse_vector< TN > & *b*, TN *correction*)**

2.2.1.2 **template<class TN> void linear_algebra::mat_sym_sparse< TN >::gen_mat (const sparse_vector< TN > & *a*, const sparse_vector< TN > & *b*, TN *correction*)** [inherited]

2.2.1.3 **mat_sparse< TN > householder (refvector < TN > & *diagonal*, refvector< TN > & *subdiagonal*)**

2.2.1.4 **mat_sparse< TN >householder (vector< TN > & *diagonal*, vector< TN > & *subdiagonal*)**

2.2.1.5 **template<class TN> mat_sparse< TN > linear_algebra::mat_sym_−sparse< TN >::householder (refvector< TN > & *diagonal*, refvector< TN > & *subdiagonal*)**[inherited]

2.2.1.6 **template<class TN> mat_sparse< TN > linear_algebra::mat_sym_−sparse< TN >::householder (vector< TN > & *diagonal*), vector< TN > & *subdiagonal*)**[inherited]

The aspired transformation is a multiplication by $(I - vv^*)$ where v is $(\pm e_1\|a_{12}\| + a_{12})/\sqrt{\pm a_{12,1}\|a_{12}\| + \|a_{12}\|^2}$. Notice that we begin counting at 0. Also see mat_sym_full::householder on the transformstions.

2.2.1.7 **template<typename TN> void std::operator>> (const TN & *a*, ostream & *OUT*)**

2.2.1.8 **template<class TN> void std::operator>> (const vector< TN > & *a*, ostream & *OUT*)**

3. Discrete Optimization in Chemical Space Namespace Documentation

3.1 Boost Namespace Reference

Classes

- class stateful_thread_group

3.2 Concepts Namespace Reference

Namespace reserved for concepts.

Classes

- class Library
  Defines the concept of an addressable set of values.
- class base_iterator
  Defines the concept of an iterator for a class of bases.
- class pruner
  Defines the concept of an addressable set of values.
- class has_gradients
  Defines the concept of a class that has gradients.
- class has_hessians
  Defines the concept of a class that has hessians.
- class has_Name
  Defines the concept of a class that has a private string Name.
- class is_threadable
  Defines the concept of a class that is threadable.
- class has_optimize
  Defines the concept of a class that has an optimize member.
- class has_stacksize
  Defines the concept of a class that has a stack member.

Functions

- template<typename T> void same_type (T const &, T const &)
  Dummy function to determine type equality.

3.2.1 Function Documentation

3.2.1.1 **template<typename T> void Concepts::same_type (T const &, T const &)**

3.3 linear_algebra Namespace Reference

Linear algebra Namespace.

Classes

- class asymmetric
  Class of antisymmetric matrices.
- class mat_asym_full
  mat_asym_full uses packing for full antisymmetric matrices.
- class mat_asym_sparse
  mat_asym_sparse packs sparse column matrices into a comfortable format.
- class mat_full
  Class of full column matrices.
- class mat_sparse
  Class of sparse column matrices.
- class mat_sym_full
  mat_sym_full uses packing for full symmetric matrices.
- class mat_sym_sparse
  mat_sym_sparse packs sparse column matrices into a comfortable format.
- class matrix
  matrix is a basic class which covers all kinds of matrices.
- class polynomial
  A class for polynomials.

- class refcount
  Class refcount provides reference counted pointers. Most basic use.

- class refvector
  Class refvector provides reference counted vectors.

- class sparse_vector
  A class for sparse vectors.

- class sparse_vector_infty
  A class for sparse vectors.

- class square
  The square matrix class contains all n by n matrices.

- class symmetric
  The symmetric class contains all symmetric matrices.

Functions

- template<class TN> mat_sym_full< TN > & (const mat_sym_full< TN > A, const mat_asym_full< TN > B, mat_sym_full< TN > &r)
  Computes the commutator AB-BA.

- template<class TN> mat_asym_full< TN > ABmBA (const mat_sym_full< TN > A, const mat_asym_full< TN > B)
  Computes the commutator AB-BA.

- template<class TN> mat_asym_full< TN > & ABmBA (const mat_sym_full< TN > B, mat_asym_full< TN > &r)
  Computes the commutator AB-BA.

- template<class TN> mat_asym_full< TN > ABmBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B)
  Computes the commutator AB-BA.

- template<class TN> mat_asym_full< TN > & ABmBA (const mat_asym_full< TN > &A, const mat_asym_full< TN > &B, mat_asym_full< TN > &r)
  Computes the commutator AB-BA.

- template<class TN> mat_asym_full< TN > ABmBA (const mat_asym_full< TN > &A, const mat_asym_full< TN > &B)
  Computes the commutator AB-BA.

- template<class TN> mat_asym_full< TN > & ABpBA (const mat_sym_full< TN > &A, const mat_asym_full< TN > &B, mat_asym_full< TN > &r)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > ABpBA (const mat_sym_full< TN > &A, const mat_asym_full< TN > &B)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > ABpBA (const mat_full< TN > &A, const mat_asym_full< TN > &B, mat_asym_full< TN > &r)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > ABpBA (const mat_full< TN > A, const mat_asym_full< TN > B)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > & ABpBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B, mat_asym_full< TN > &r)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > ABpBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > & ABpBA (const mat_sym_full< TN > A, const mat_asym_sparse< TN > B, mat_asym_full< TN > &r)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > ABpBA (const mat_sym_full< TN > A, const mat_asym_sparse< TN > B)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > & ABpBA (const mat_asym_sparse< TN > A, const mat_sym_full< TN > B,mat_asym_full< TN > &r)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > ABpBA (const mat_asym_sparse< TN > A, const mat_sym_full< TN > B)
  Computes the anti-commutator AB+BA.

- template<class TN>mat_sym_full< TN > & ABpBA (const mat_sym_full< TN > A, mat_sym_full< TN > B, mat_sym_full< TN > &r, mat_sym_full< TN > &I, mat_full< TN > &I2)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_sym_full< TN > & ABpBA (const mat_sym_full< TN > A, mat_sym_full< TN > B, mat_sym_full< TN > &r)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_sym_full< TN > & ABpBA (const mat_sym_full< TN > &A, const mat_sym_full< TN > &B, mat_sym_full< TN > &r, mat_sym_full< TN > &I)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_sym_full< TN > ABpBA (const mat_sym_full< TN > &A, const mat_sym_full< TN > &B)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_sym_full< TN > & ABpBA (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_sym_full< TN > &r, mat_full< TN > &I2)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_sym_full< TN > & ABpBA (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_sym_full< TN > &r)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_sym_full< TN > ABpBA (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > & ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_full< TN > &r, mat_full< TN > &I2)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_sparse< TN > & ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_sparse< TN > &r, mat_sparse< TN > &I2)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > & ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_full< TN > &r, mat_sparse< TN > &I2)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > & ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_full< TN > &r)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_sparse< TN > & ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_sparse< TN > &r)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B)
  Computes the anti-commutator AB+BA.

- template<class T> mat_sym_full< T > ABpBA (const mat_asym_full< T > &A, const mat_asym_full< T > &B)
  Commutes the anti-commutator AB+BA. Lame and lazy version.

- template<class TN> matrix< TN > & zero (matrix< TN > &r)

- template<class TN> mat_full< TN > & zero (mat_full< TN > &r)

- template<class TN> mat_sym_full< TN > & zero (mat_sym_full< TN > &r)

- template<class TN> mat_asym_full< TN > & zero (mat_asym_full< TN > &r)

- template<class TN> refvector< TN > & zero (refvector< TN > &r)

- double & zero (double &r)

- double & one (double &r)

- template<class TN> matrix< TN > & one (matrix< TN > &r)

- void copy (double &a, const double &b, const long i)

- void copy (double &a, const double &b)

- void copy (long &a, const long &b, const long i)

- void copy (long &a, const long &b)

- template<class TN> void copy (mat_sym_sparse< TN > &a, mat_sym_sparse< TN > &b, const long i)

- template<class TN> void copy (mat_sym_sparse< TN > &a, const mat_sym_sparse< TN > &b)

- template<class TN> void copy (mat_sym_sparse< TN > &a, mat_sym_sparse< TN > &b)

- template<class TN> void copy (mat_sym_full< TN > &a, mat_sym_full< TN > &b, const long i)

- template<class TN> void copy (mat_sym_full< TN > &a, const mat_sym_full< TN > &b)

- template<class TN> void copy (mat_sym_full< TN > &a, mat_sym_full< TN > &b)

- template<class TN> void copy (mat_full< TN > &a, mat_full< TN > &b, const long i)

- template<class TN> void copy (mat_full< TN > &a, const mat_full< TN > &b)

- template<class TN> void copy (mat_full< TN > &a, mat_full< TN > &b)

- template<class TN> void copy (mat_sparse< TN > &a, mat_sparse< TN > &b, const long i)

- template<class TN> void copy (mat_sparse< TN > &a, const mat_sparse< TN > &b)

- template<class TN> void copy (mat_sparse< TN > &a, mat_sparse< TN > &b)

- template<class TN> void copy (refvector< TN > &a, refvector< TN > &b, const long i)
- template<class TN> void copy (refvector< TN > &a, const refvector< TN > &b)
- template<class TN> void copy (refvector< TN > &a, refvector< TN > &b)
- template<class TN> void copy (sparse_vector< TN > &a, sparse_vector< TN > &b, const long i)
- template<class TN> void copy (sparse_vector< TN > &a, const sparse_vector< TN > &b)
- template<class C> void copy (C &a, const C &b)
- template<class TN> void display (const matrix< TN > &a, ostream &outs)
- template<class TN> void display (const refvector< TN > &a, ostream &outs)
- void display (const double &a, ostream &outs)
- template<class TNTN> mat_full< TNTN > exp (const mat_asym_full< TNTN > &A)
- template<class TNTN> mat_full< TNTN > dexp (mat_asym_full< TNTN > A, const long k, const long l)
- template<class TN> mat_sym_sparse< TN > & zero (mat_sym_sparse< TN > &r)
- template<class TN> mat_sparse< TN > & zero (mat_sparse< TN > &r)
- template<typename TN> TN & zero (TN &r)

- void copy (bool &a, const bool &b)

- template<class TN> void copy (mat_full< TN > &a, const mat_full< TN > &b, const long i)

- template<class TN> void copy (mat_sparse< TN > &a, const mat_sparse< TN > &b, const long i)

- template<class TN> void copy (mat_sym_sparse< TN > &a, const mat_sym_sparse< TN > &b, const long i)

- template<class TN> void copy (mat_sym_full< TN > &a, const mat_sym_full< TN > &b, const long i)

- template<class TN> void copy (refvector< TN > &a, const refvector< TN > &b, const long i)

- template<class TN> TN operator * (const std::vector< TN > &a, const std::vector< TN > &b)
  Scalar product of two full vectors.

- template<class TN> TN operator * (const vector< TN > &a, const vector< TN > &b)
  Scalar product of two full vectors.

- template<class TN> TN operator * (sparse_vector< TN > a, std::vector< TN > b)
  Scalar product of a sparse_vector with a full vector.

- template<class TN> TN operator * (sparse_vector< TN > a, refvector< TN > b)
  Scalar product of a sparse_vector with a full refvector.

- template<class TN> TN operator * (sparse_vector< TN > a, vector< TN > b)
  Scalar product of a sparse_vector with a full vector.

- template<class TN> TN trace_AB (const mat_sym_full< TN > &A, const mat_sym_full< TN > &B)

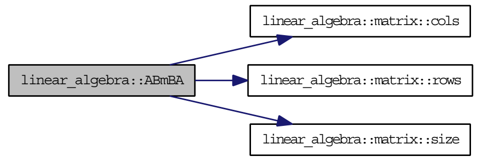
- template<class TN> TN trace_AB (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B)

- template<class T> T trace_AB (const mat_asym_full< T > &A, const mat_asym_full< T > &B)

3.3.1 Function Documentation

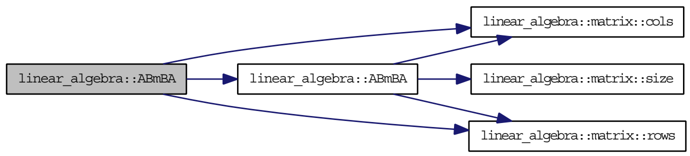
3.3.1.1 template<class TN> mat_sym_full<TN>& linear_algebra::ABmBA (const mat_sym_full< TN > *A*, const mat_asym_full< TN > *B*, mat_sym_full< TN > & *r*)

Here is the call graph for this function:



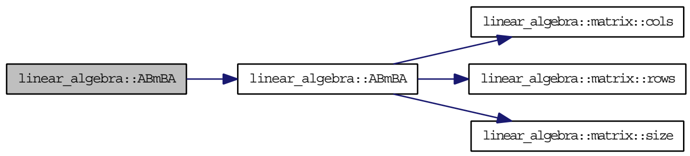
3.3.1.2 template<class TN> mat_asym_full<TN> linear_algebra::ABmBA (const mat_sym_full< TN > *A*, const mat_asym_full< TN > *B*)

Here is the call graph for this function:



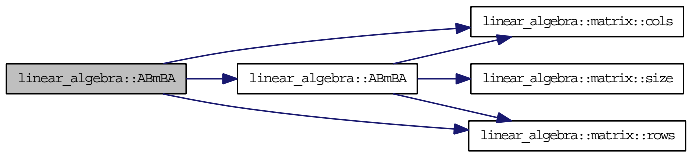
3.3.1.3 template<class TN> mat_asym_full<TN>& linear_algebra::ABmBA (const mat_asym_full< TN > *A*, const mat_sym_full< TN > *B*, mat_asym_full< TN > & *r*)

Here is the call graph for this function:



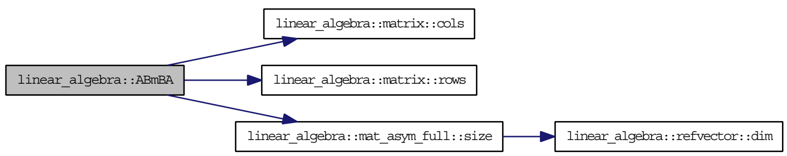
3.3.1.4 template<class TN> mat_asym_full<TN> linear_algebra::ABmBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B)

Here is the call graph for this function:



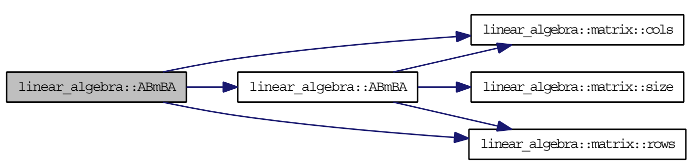
3.3.1.5 template<class TN> mat_asym_full<TN>& linear_algebra::ABmBA (const mat_asym_full< TN > & A, const mat_asym_full< TN > & B, mat_asym_full< TN > & r)

Here is the call graph for this function:



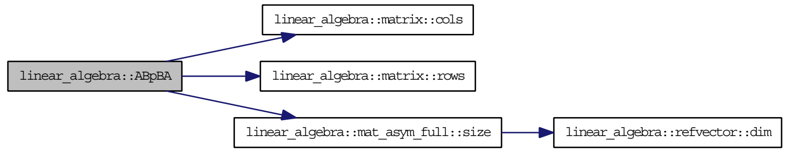
3.3.1.6 template<class TN> mat_asym_full<TN> linear_algebra::ABmBA (const mat_asym_full< TN > & A, const mat_asym_full< TN > & B)

Here is the call graph for this function:



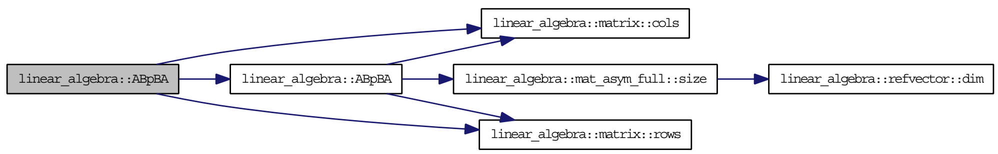
3.3.1.7 template<class TN> mat_asym_full<TN>& linear_algebra::ABpBA (const mat_sym_full< TN > & A, const mat_asym_full< TN > & B, mat_asym_full< TN > & r)

Here is the call graph for this function:



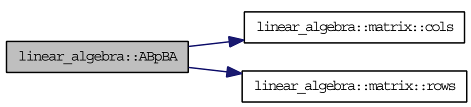
3.3.1.8 template<class TN> mat_asym_full<TN> linear_algebra::ABpBA (const mat_sym_full< TN > & A, const mat_asym_full< TN > & B—)

Here is the call graph for this function:



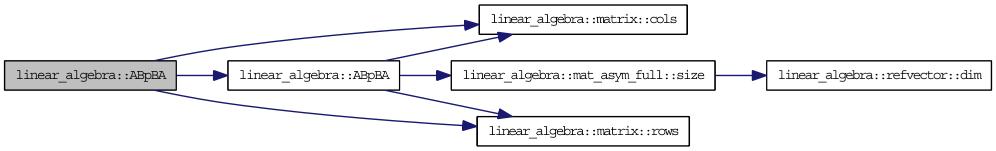
3.3.1.9 template<class TN> mat_asym_full<TN> linear_algebra::ABpBA (const mat_full< TN > & A, const mat_asym_full< TN > & B, mat_asym_full< TN > & r)

Here is the call graph for this function:



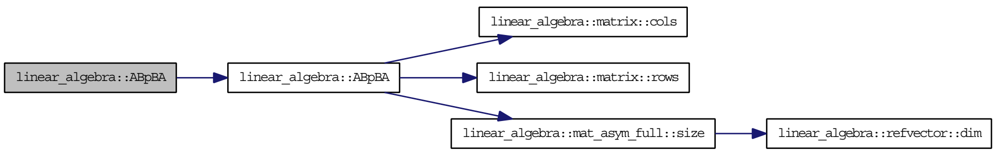
3.3.1.10 template<class TN> mat_asym_full<TN> linear_algebra::ABpBA (const mat_full< TN > A, const mat_asym_full< TN > B)

Here is the call graph for this function:



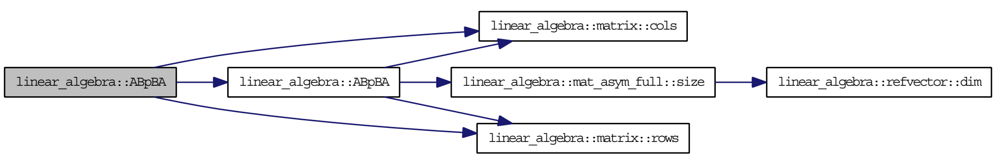
3.3.1.11 template<class TN> mat_asym_full<TN>& linear_algebra::ABpBA (const mat_asym_full< TN > *A*, const mat_sym_full< TN > *B*, mat_asym_full< TN > & *r*)

Here is the call graph for this function:



3.3.1.12 template<class TN> mat_asym_full<TN> linear_algebra::ABpBA (const mat_asym_full< TN > *A*, const mat_sym_full< TN > *B*)

Here is the call graph for this function:

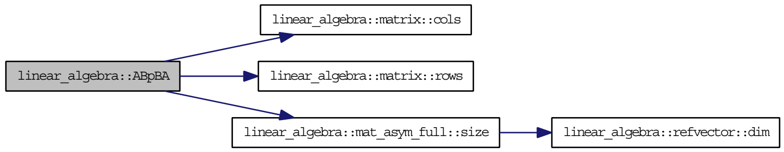


3.3.1.13 template<class TN> mat4_4asym4_4full<TN>& linear_algebra::ABpBA (const mat_sym_full< TN > *A*, const mat_asym_sparse< TN > *B*, mat_asym_full< TN > & *r*)

Todo

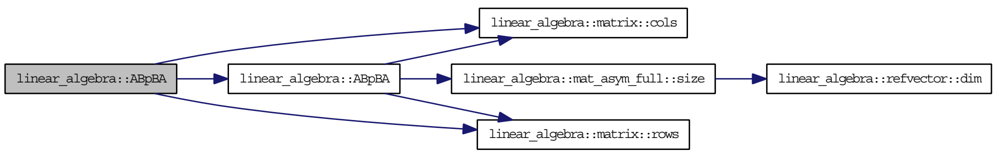
Need optimization taking advantage of sparse structure.

Here is the call graph for this function:



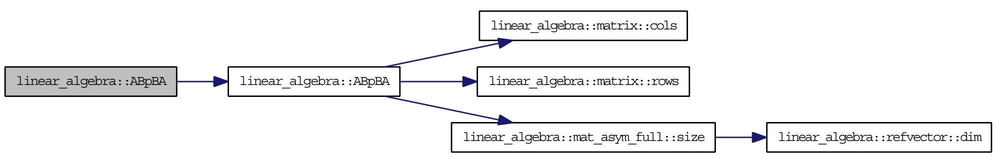
3.3.1.14 template<class TN> mat_asym_full<TN> linear_algebra::ABpBA (const mat_sym_full< TN > A, const mat_asym_sparse< TN > B)

Here is the call graph for this function:



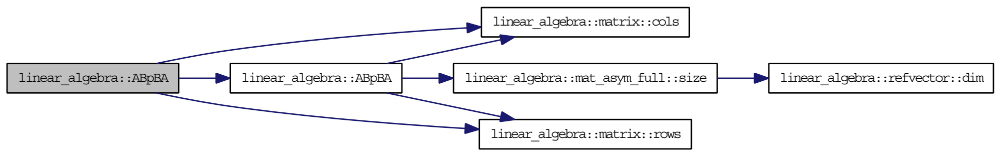
3.3.1.15 template<class TN> mat_asym_full<TN>& linear_algebra::ABpBA (const mat_asym_sparse< TN > A, const mat_sym_full< TN > B, mat_asym_full< TN > & r)

Here is the call graph for this function:



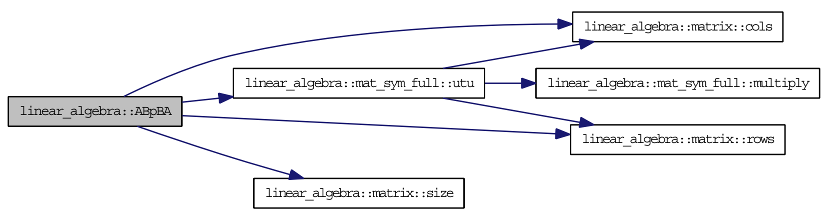
3.3.1.16 template<class TN> mat_asym_full<TN> linear_algebra::ABpBA (const mat_asym_sparse< TN > A, const mat_sym_full< TN > B)

Here is the call graph for this function:



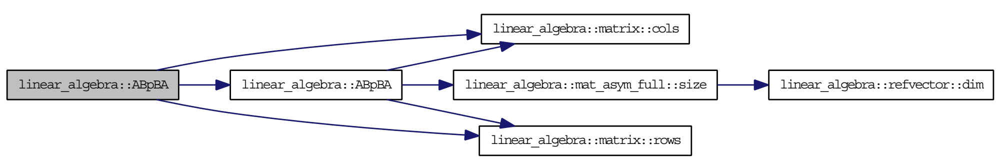
3.3.1.17 template<class TN> mat_sym_full<TN>& linear_algebra::ABpBA (const mat_sym_full< TN > *A*, mat_sym_full< TN > *B*, mat_sym_full< TN > & *r*, mat_sym_full< TN > & *I*, mat_full< TN > & *I2*)

Here is the call graph for this function:



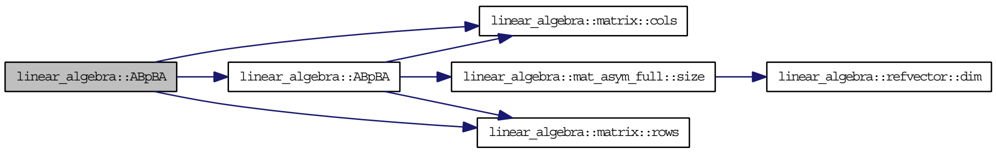
3.3.1.18 template<class TN> mat_sym_full<TN>& linear_algebra::ABpBA (const mat_sym_full< TN > *A*, mat_sym_full< TN > *B*, mat_sym_full< TN > & *r*)

Here is the call graph for this function:



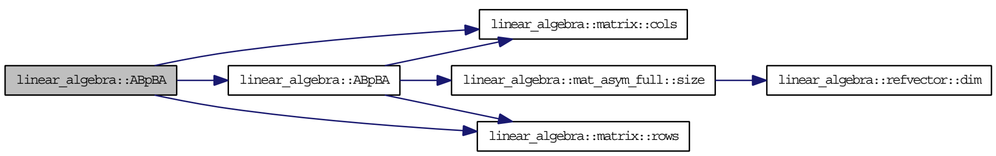
3.3.19 template<class TN> mat_sym_full<TN>& linear_algebra::ABpBA (const mat_sym_full< TN > & A, const mat_sym_full< TN > & B, mat_sym_full< TN > & r, mat_sym_full< TN > & I)

Here is the call graph for this function:



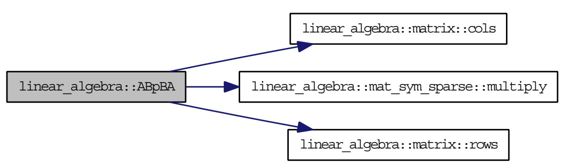
3.3.1.20 template<class TN> mat_sym_full<TN> linear_algebra::ABpBA (const mat_sym_full< TN > & A, const mat_sym_full< TN > & B)

Here is the call graph for this function:



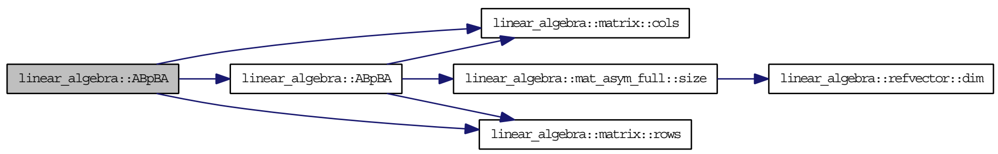
3.3.1.21 template<class TN> mat_sym_full<TN>& linear_algebra::ABpBA (const mat_sym_full< TN > & A, const mat_sym_sparse< TN > & B, mat_sym_full< TN > & r, mat_full< TN > & $I2$)

Here is the call graph for this function:



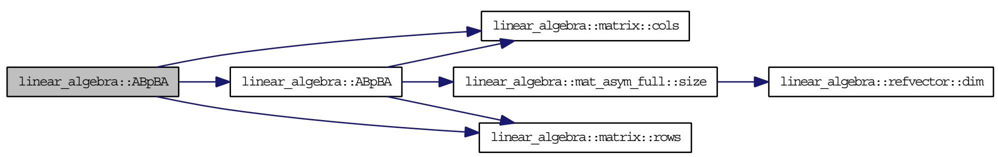
3.3.1.22 template<class TN> mat_sym_full<TN>& linear_algebra::ABpBA (const mat_sym_full< TN > & *A*, const mat_sym_sparse< TN > & *B*, mat_sym_full< TN > & *r*)

Here is the call graph for this function:



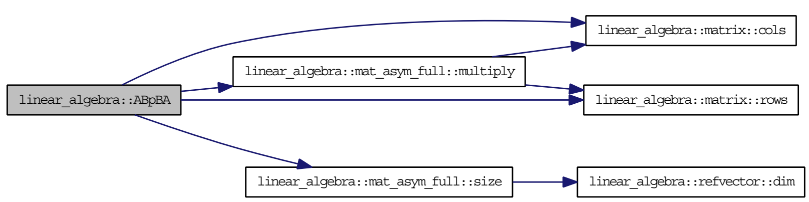
3.3.1.23 template<class TN> mat_sym_full<TN> linear_algebra::ABpBA (const mat_sym_full< TN > & *A*, const mat_sym_sparse< TN > & *B*)

Here is the call graph for this function:



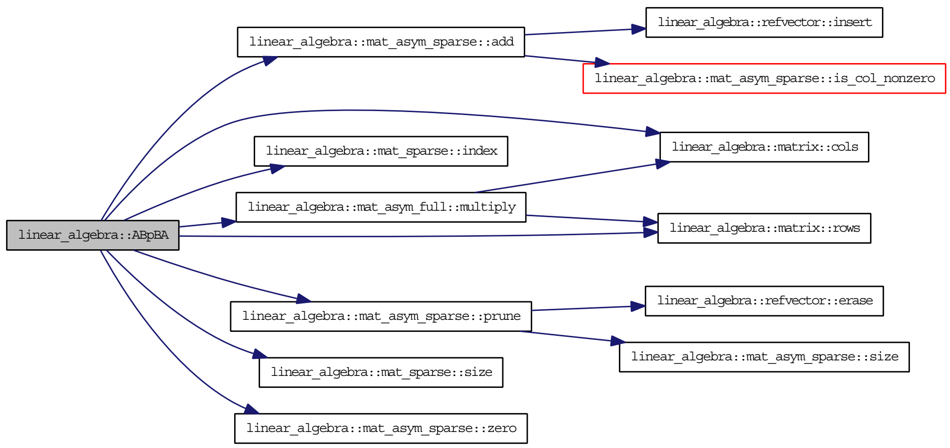
3.3.1.24 template<class TN> mat_asym_full<TN>& linear_algebra::ABpBA (const mat_asym_full< TN > & *A*, const mat_sym_sparse< TN > & *B*, mat_asym_full< TN > & *r*, mat_full< TN > & *[I2*)

Here is the call graph for this function:



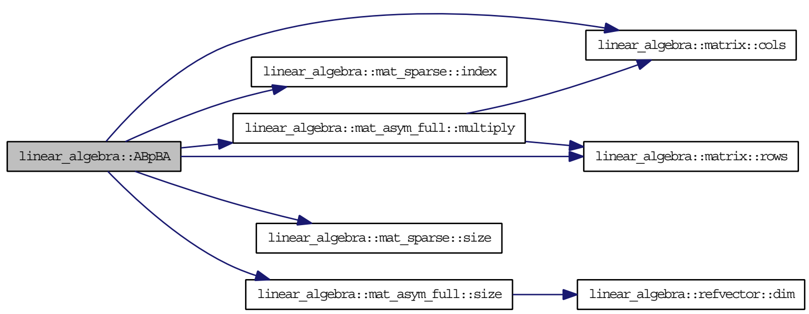
3.3.1.25 template<class TN> mat_asym_sparse<TN>& linear_algebra::ABpBA (const mat_asym_full< TN > & A, const mat_sym_sparse< TN > & B, mat_asym_sparse< TN > & r, mat_sparse< TN > & $I2$)

Here is the call graph for this function:



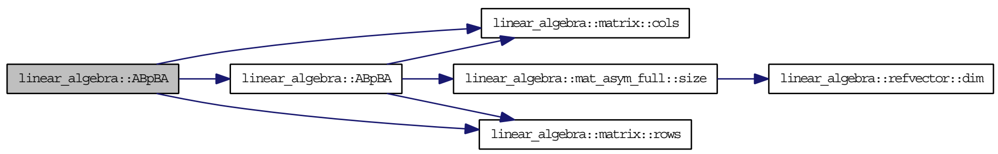
3.3.1.26 template<class TN> mat_asym_full<TN>& linear_algebra::ABpBA (const mat_asym_full< TN > & A, const mat_sym_sparse< TN > & B, mat_asym_full< TN > & r, mat_sparse< TN > & $I2$)

Here is the call graph for this function:



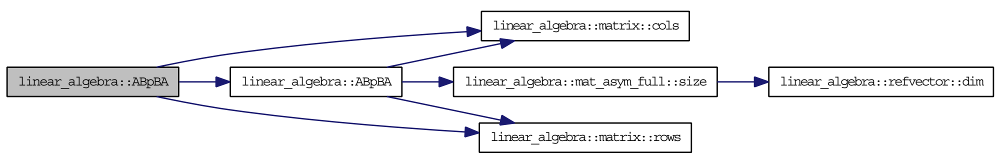
3.3.1.27 template<class TN> mat_asym_full<TN>& linear_algebra::ABpBA (const mat_asym_full< TN > & A, const mat_sym_sparse< TN > & B, mat_asym_full< TN > & r)

Here is the call graph for this function:



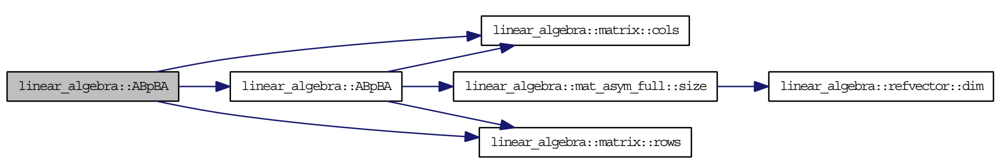
3.3.1.28 template<class TN> mat_asym_sparse<TN>& linear_algebra::ABpBA (const mat_asym_full< TN > & A, const mat_sym_sparse< TN > & B, mat_asym_sparse< TN > & r)

Here is the call graph for this function:



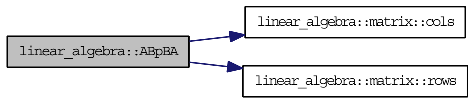
3.3.1.29 template<class TN> mat_asym_full<TN> linear_algebra::ABpBA (const mat_asym_full< TN > & A, const mat_sym_sparse< TN > & B)

Here is the call graph for this function:



3.3.1.30 template<class T> mat_sym_full<T> linear_algebra::ABpBA (const mat_asym_full< T > & A, const mat_asym_full< T > & B)

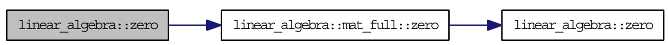
Here is the call graph for this function:



3.3.1.31 template<class TN> matrix<TN>& linear_algebra::zero (matrix< TN > & r)[inline]

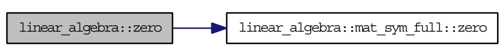
3.3.1.32 template<class TN> mat_full< TN > & linear_algebra::zero (mat_full< TN > & r)[inline]

Here is the call graph for this function:



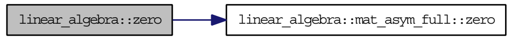
3.3.1.33 template<class TN> mat_sym_full< TN > & linear_algebra::zero (mat_sym_full< TN > & r)[inline]

Here is the call graph for this function:



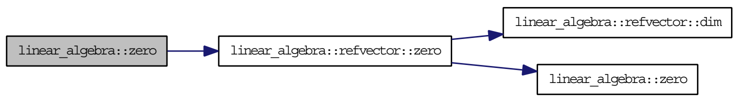
3.3.1.34 template<class TN> mat_asym_full< TN > & linear_algebra::zero (mat_asym_full< TN > & r)[inline]

Here is the call graph for this function:



3.3.1.35 template<class TN> refvector< TN > & linear_algebra::zero (refvector< TN > & r)[inline]

Here is the call graph for this function:



3.3.1.36 double & linear_algebra::zero (double & r)[inline]

3.3.1.37 double & linear_algebra::one (double & r)[inline]

3.3.1.38 template<class TN> matrix< TN > & linear_algebra::one (matrix< TN > & r)[inline]

3.3.1.39 void linear_algebra::copy (double & a, const double & b, const long i)[inline]

3.3.1.40 void linear_algebra::copy (double & a, const double & b)[inline]

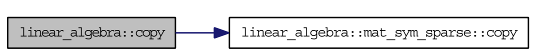
3.3.1.41 void linear_algebra::copy (long & a, const long & b, const long i)[inline]

3.3.1.42 void linear_algebra::copy (long & a, const long & b)[inline]

3.3.1.43 template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > & a, mat_sym_sparse< TN > & b, const long i)tt [inline]

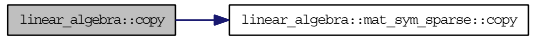
3.3.1.44 template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > & a, const mat_sym_sparse< TN > & b[inline]

Here is the call graph for this function:



3.3.1.45 template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > & a, mata_sym_sparse< TN > & b)[inline]

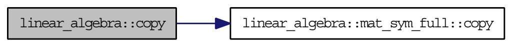
Here is the call graph for this function:



3.3.1.46 template<class TN> void linear_algebra::copy (mat_sym_full< TN > & a, mat_sym_full< TN > & b, const long i)[inline]

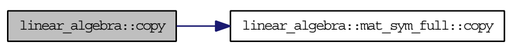
3.3.1.47 template<class TN> void linear_algebra::copy (mat_sym_full< TN > & a, const mat_sym_full< TN > & b)[inline]

Here is the call graph for this function:



3.3.1.48 template<class TN> void linear_algebra::copy (mat_sym_full< TN > & a, mat_sym_full< TN > & b)[inline]

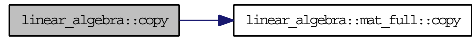
Here is the call graph for this function:



3.3.1.49 template<class TN> void linear_algebra::copy (mat_full< TN > & a, mat_full< TN > & b, const long i)[inline]

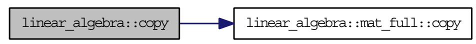
3.3.1.50 template<class TN> void linear_algebra::copy (mat_full< TN > & a, const mat_full< TN > & b)[inline]

Here is the call graph for this function:



3.3.1.51 template<class TN> void linear_algebra::copy (mat_full< TN > & a, mat_full< TN > & b)[inline]

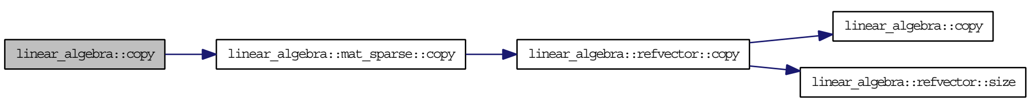
Here is the call graph for this function:



3.3.1.52 template<class TN> void linear_algebra::copy (mat_sparse< TN > & a, mat_sparse< TN > & b, const long i)[inline]

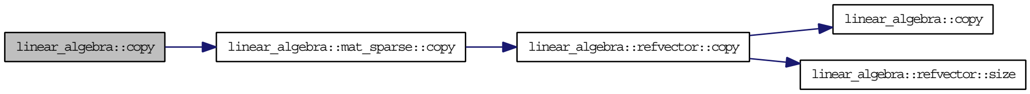
3.3.1.53 template<class TN> void linear_algebra::copy (mat_sparse< TN > & a, const mat_sparse< TN > & b)[inline]

Here is the call graph for this function:



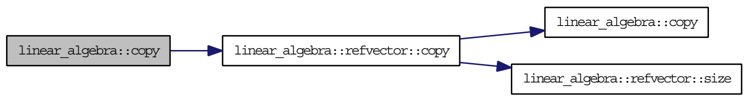
3.3.1.54 template<class TN> void linear_algebra::copy (mat_sparse< TN > & a, mat_sparse< TN > & b)[inline]

Here is the call graph for this function:



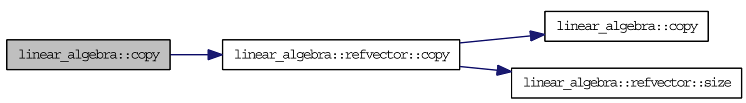
3.3.1.55 template<class TN> void linear_algebra::copy (refvector< TN > & a, refvector< TN > & b, const long i)[inline]

Here is the call graph for this function:



3.3.1.56 template<class TN> void linear_algebra::copy (refvector< TN > & a, const refvector< TN > & b){[inline]

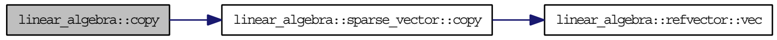
Here is the call graph for this function:



3.3.1.57 template<class TN> void linear_algebra::copy (refvector< TN > & a, refvector< TN > & b){*[inline]*

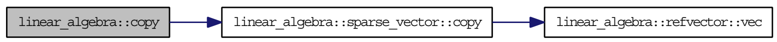
3.3.1.58 template<class TN> void linear_algebra::copy (sparse_vector< TN > & a, sparse_vector< TN > & b, const long i)[inline]

Here is the call graph for this function:



3.3.1.59 template<class TN> void linear_algebra::copy (sparse_vector< TN > & *a*, const sparse_vector< TN > & *b*)[inline]

Here is the call graph for this function:

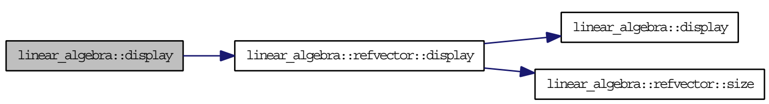


3.3.1.60 template<class C> void linear_algebra::copy (C & *a*, const C & *b*)[inline]

3.3.1.61 template<class TN> void linear_algebra::display (const matrix< TN > & *a*, ostream & *outs*)[inline]

3.3.1.62 template<class TN> void linear_algebra::display (const refvector< TN > & *a*, ostream & *outs*)[inline]

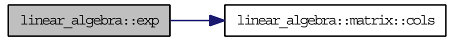
Here is the call graph for this function:



3.3.1.63 void linear_algebra::display (const double & *a*, ostream & *outs*)[inline]

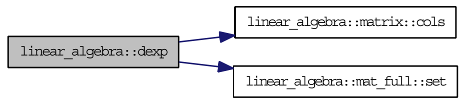
3.3.1.64 template<class TNTN> mat_full< TNTN > linear_algebra::exp (const mat_asym_full< TNTN > & *A*)

Here is the call graph for this function:



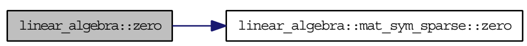
3.3.1.65 template<class TNTN> mat_full< TNTN > linear_algebra::dexp (mat_asym_full< TNTN > A, const long k, const long l)

Here is the call graph for this function:



3.3.1.66 template<class TN> mat_sym_sparse<TN>& linear_algebra::zero (mat_sym_sparse< TN > & r)[inline]

Here is the call graph for this function:



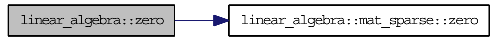
3.3.1.67 template<class TN> mat_sparse<TN>& linear_algebra::zero (mat_sparse< TN > & r)[inline]

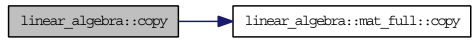
3.3.1.68 template<typename TN> TN& linear_algebra::zero (TN & r)[inline]

3.3.1.69 void linear_algebra::copy (bool & a, const bool & b)[inline]

3.3.1.70 template<class TN> void linear_algebra::copy (mat_full< TN > & a, const mat_full< TN > & b, const long i){[inline]

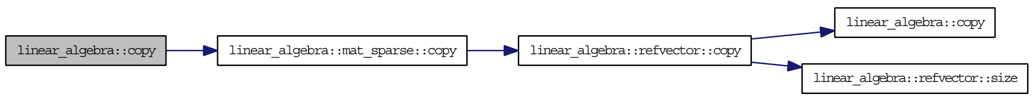
Here is the call graph for this function:





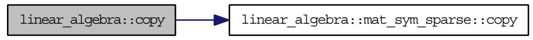
3.3.1.71 template<class TN> void linear_algebra::copy (mat_sparse< TN > & a, const mat_sparse< TN > & *b*, const long *i*)[inline]

Here is the call graph for this function:



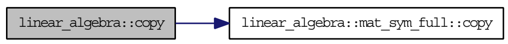
3.3.1.72 template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > & *a*, const mat_sym_sparse< TN > & *b*, const long *i*)[inline]

Here is the call graph for this function:



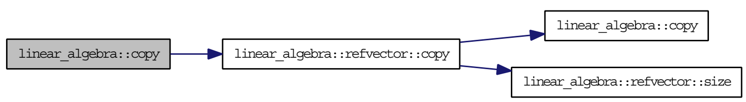
3.3.1.73 template<class TN> void linear_algebra::copy (mat_sym_full< TN > & a, const mat_sym_full< TN > & b, const long i)[inline]

Here is the call graph for this function:



3.3.1.74 template<class TN> void linear_algebra::copy (refvector< TN > & a, const refvector< TN > & b, const long i)[inline]

Here is the call graph for this function:



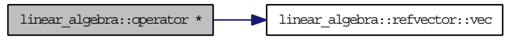
3.3.1.75 template<class TN> TN linear_algebra::operator * (const std::vector< TN > & a, const std::vector< TN > & b)

3.3.1.76 template<class TN> TN linear_algebra::operator * (const vector< TN > & a, const vector< TN > & b)

3.3.1.77 template<class TN> TN linear_algebra::operator * (sparse_vector< TN > a, std::vector< TN > b)[inline]

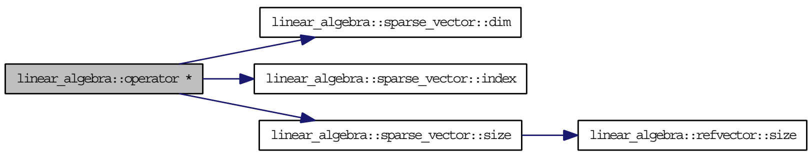
3.3.1.78 template<class TN> TN linear_algebra::operator * (sparse_vector< TN > a, refvector< TN > b)[inline]

Here is the call graph for this function:



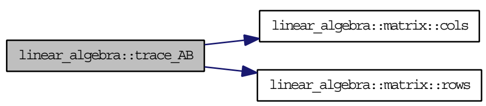
3.3.1.79 template<class TN> TN linear_algebra::operator * (sparse_vector< TN > a, vector< TN > b)[inline]

Here is the call graph for this function:



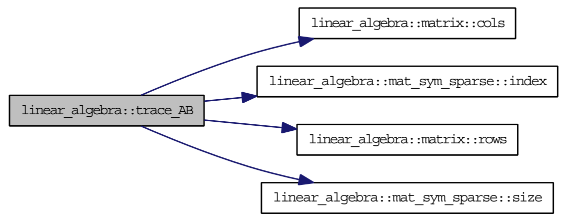
3.3.1.80 template<class TN> TN linear_algebra::trace_AB (const mat_sym_full< TN > & A, const mat_sym_full< TN > & B)

Here is the call graph for this function:



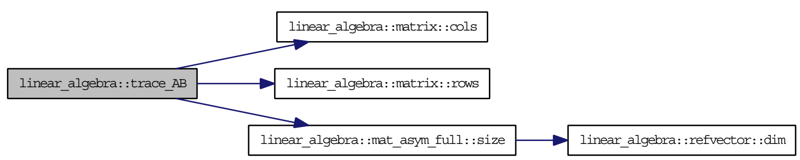
3.3.1.81 template<class TN> TN linear_algebra::trace_AB (const mat_sym_full< TN > & A, const mat_sym_sparse< TN > & B)

Here is the call graph for this function:



3.3.1.82 template<class T> T linear_algebra::trace_AB (const mat_asym_full< T > & *A*, const mat_asym_full< T > & *B*)

Here is the call graph for this function:



3.4 std Namespace Reference

Functions

- template<class TN> void (const vector< TN > &a, ostream &OUT)
- template<typename TN> void (const TN &a, ostream &OUT)

4. Discrete Optimization in Chemical Space Class Documentation

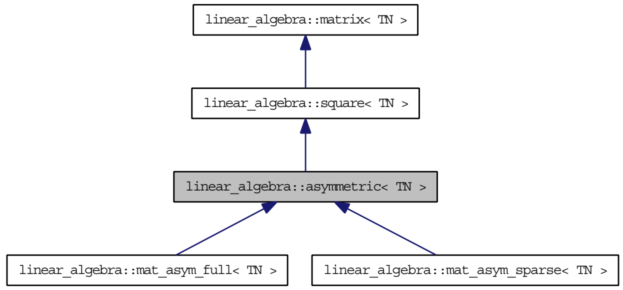
4.1 linear_algebra::asymmetric< TN > Class Template Reference

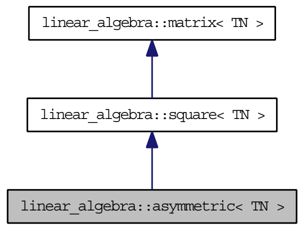
Class of antisymmetric matrices.

Inheritance diagram for linear_algebra::asymmetric< TN >:

Collaboration diagram for linear_algebra::asymmetric< TN >:

Public Member Functions





- asymmetric< TN > & transpose ()
- asymmetric< TN > operator ∗ (const TN a) const

template<class TN> class linear_algebra::asymmetric< TN >

4.1.1 Member Function Documentation

4.1.1.1 template<class TN> asymmetric< TN > linear_algebra::asymmetric< TN >::operator ∗ (const TN *a*) const

Reimplemented in linear_algebra::mat_asym_full< TN >.

4.1.1.2 template<class TN> asymmetric< TN > & linear_algebra::asymmetric< TN >::transpose ()

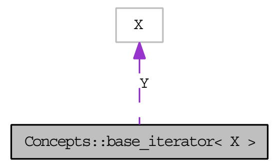
The documentation for this class was generated from the following files:

- include/BCR_CPP_LA/asymmetric.decl
- include/BCR_CPP_LA/asymmetric.h

4.2 Concepts::base_iterator< X > Class Template Reference

Concept defines an iterator for a class of bases.

`#include <concepts.hh>` Collaboration diagram for Concepts::base_iterator< X >:



Public Member Functions

- BOOST_CONCEPT_USAGE (base_iterator)

Private Attributes

- X & Y

template<class X> class Concepts::base_iterator< X >

4.2.1 Member Function Documentation

4.2.1.1 template<class X> Concepts::base_iterator< X >::BOOST_CONCEPT_USAGE
base_iterator< X >)
[inline]

Here is the call graph for this function:

4.2.2 Member Data Documentation

4.2.2.1 template<class X> X& Concepts::base_iterator< X >::Y[private]

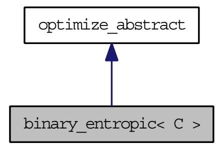
The documentation for this class was generated from the following file:

- concepts.hh

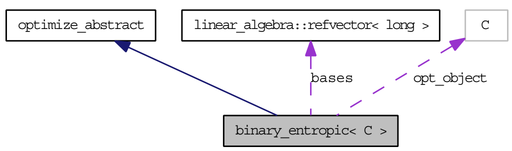
4.3 binary_entropic< C > Class Template Reference

Class for enhanced sampling using an entropic measure of coverage.

#include <binary_entropic.hh> Inheritance diagram for binary_entropic< C >:



Collaboration diagram for binary_entropic< C >:



Public Member Functions

- binary_entropic (const C &a)
  Copy constructor.
- ulong optimize (ulong N) const
  Meta-Optimize by generating maximally distant starting configurations.
- valerg get_value (const ulong i) const

Public Attributes

- ulong nruns
  Number of runs.
- long max_steps
  Maximum number of steps.

Protected Attributes

- C opt_object
  This is an optimization object of type C. This is the underlying optimization method.

- refvector< long > bases
  This vector defines the substitution bases used in the optimization and entropic maximization (maximize_entropic_distance()).

Private Member Functions

- BOOST_CONCEPT_ASSERT ((Concepts::has_optimize< C >))
  Require C to conform to an optimization.
- BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))
  Require C to conform to the pruner concept.

4.3.1 Detailed Description

template<class C> class binary_entropic< C >

This class provides a meta-optimization mechanism. The secondary optimizer C has to comply with the Concepts::has_optimize and Concepts::pruner concepts in order for the optimization to work. After each successful optimization the visited configurations are analysed for coverage of the library. maximize_entropic_distance() is then used to determine the next best starting point.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 template<class C> binary_entropic< C >::binary_entropic (const C & *a*)[inline]

4.3.3 Member Function Documentation

4.3.3.1 template<class C> binary_entropic< C >::BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >)) [private]

4.3.3.2 template<class C> binary_entropic< C >::BOOST_CONCEPT_ASSERT ((Concepts::has_optimize< C >)) [private]

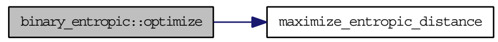
4.3.3.3 template<class C> valerg binary_entropic< C >::get_value (const ulong *i*) const [inline]

4.3.3.4 template<class C> ulong binary_entropic< C >::optimize (ulong *N*) const [inline, virtual]

The metric used to determine the distance is implemented and explained in maximize_entropic_distance().

Implements optimize_abstract.

Here is the call graph for this function:



4.3.4 Member Data Documentation

4.3.4.1 template<class C> refvector<long> binary_entropic< C >::bases[protected]

4.3.4.2 template<class C> long binary_entropic< C >::max_steps

4.3.4.3 template<class C>ulong binary_entropic< C >::nruns

4.3.4.4 template<class C> C binary_entropic< C >::opt_object[protected]

The documentation for this class was generated from the following file:

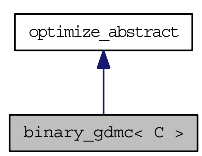
- binary_entropic.hh

4.4 binary_gdmc< C > Class Template Reference

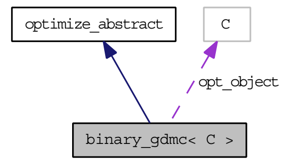
Gradient-directed Monte Carlo performed on a hypercube.

#include <binarygdmc.hh>

Inheritance diagram for binary_gdmc< C >:



Collaboration diagram for binary_gdmc< C >:



Public Member Functions

- binary_gdmc (const C &a)
  Copy Constructor.
- ulong optimize (ulong N) const
  Gradient-directed Monte-Carlo optimization.
- valerg get_value (const ulong i) const
  Return the value for molecule i.

Public Attributes

- double T
  Temperature-analogue to be used for Monte-Carlo evaluations.
- ulong tight_steps
  Number of tightening steps. Determines the strictness of acceptance.
- ulong max_steps
  Maximum number of steps.

Private Member Functions

- BOOST_CONCEPT_ASSERT ((Concepts::has_gradients< C >))
  Require C to conform to the Concepts::has_gradients concept.
- BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))
  Require C to conform to the Concepts::pruner concept.
- BOOST_CONCEPT_ASSERT ((Concepts::has_optimize< C >))
  Require C to conform to an optimization.

Private Attributes

- C opt_object
  This is the library object to be used in the optimization.

4.4.1 Detailed Description

template<class C> class binary_gdmc< C > This class takes a Library object and performs an optimization. The Library is arranged in a hypercube topology in which each molecule occupies a vertex on the hull of the cube. After a local optimum has been reached, a Monte-Carlo simulation is performed to determine a new starting point. Each time a new minimum is found the acceptance criteria become more strict.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 template<class C> binary_gdmc< C >::binary_gdmc (const C & *a*)[inline]

4.4.3 Member Function Documentation

4.4.3.1 template<class C> binary_gdmc< C >::BOOST_CONCEPT_ASSERT ((Concepts::has_optimize< C >))[private]

4.4.3.2 template<class C> binary_gdmc< C >::BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))[private]

4.4.3.3 template<class C> binary_gdmc< C >::BOOST_CONCEPT_ASSERT ((Concepts::has_gradients< C >))[private]

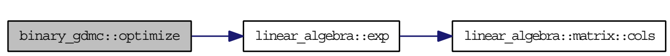
4.4.3.4 template<class C> valerg binary_gdmc< C >::get_value (const ulong *i*) const[inline]

4.4.3.5 template<class C> ulong binary_gdmc< C >::optimize (ulong *N*) const[inline, virtual]

The gradient to be used is determined by the gradient() method of the class C.

Implements optimize_abstract.

Here is the call graph for this function:



4.4.4 Member Data Documentation

4.4.4.1 template<class C> ulong binary_gdmc< C >::max_steps

4.4.4.2 template<class C> C binary_gdmc< C >::opt_object[private]

4.4.4.3 template<class C> double binary_gdmc< C >::T

4.4.4.4 template<class C> ulong binary_gdmc< C >::tight_steps

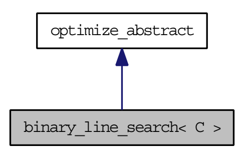
The documentation for this class was generated from the following file:

- binarygdmc.hh

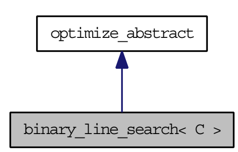
4.5 binary_line_search< C > Class Template Reference

Search a pruned Library using a linesearch algorithm.

#include <binary_line_search.hh> Inheritance diagram for binary_line_search< C >:



Collaboration diagram for binary_line_search< C >:



Public Member Functions

- binary_line_search ()
  Default constructor.
- binary_line_search (const binary_line_search &a)
  Copy constructor.

- ulong optimize (ulong N) const
  Default optimization with a clear history and no pruning in the first iteration.
- ulong optimize (ulong N, double lambda, refvector< ulong > &visited_run) const
  Optimize starting from conformation N.

Private Member Functions

- BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))
  Require Concepts::pruner compliance of C.
- BOOST_CONCEPT_ASSERT ((boost::DefaultConstructible< C >))

4.5.1 Detailed Description

template<class C> class binary_line_search< C >

In this method the Library is spanned *v*ia a hypercube. Each molecule in the library sits on one vertex of the cube. This leads to a bit representation of each molecule. The line search is then performed on the bits.

Constructor & Destructor Documentation

4.5.1.1 template<class C> binary_line_search< C >::binary_line_search ()[inline]

4.5.1.2 template<class C> binary_line_search< C >::binary_line_search (const binary_line_search< C > & *a*)

4.5.2 Member Function Documentation

4.5.2.1 template<class C> binary_line_search< C >::BOOST_CONCEPT_ASSERT ((boost::DefaultConstructible< C >))[private]

4.5.2.2 template<class C> binary_line_search< C >::BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))[private]

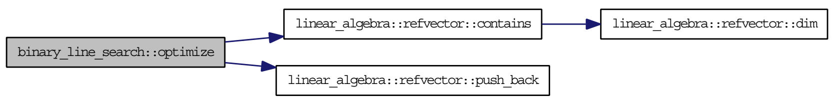
4.5.2.3 template<class C> ulong binary_line_search< C >::optimize (ulong *N*, double *lambda*, refvector< ulong > & *visited_run*) const[inline]

A Lagrange multiplier method is employed to enforce boundary constraints. The multiplier increases as the optimization proceeds, enforcing the constraint ever more rigorously.

In essence we are following an interior point method, in which we reestablish feasibility every log(library size) cycles. The precise rules for pruning and adjusting the Lagrange multiplier can be found in prune(). At each position in the bit string representing the reference molecule N, the molecule with that bit flipped from N is computed and compared. If there is an improvement in the objective function $P + \lambda\pi$, where P is the property, π is the penalty and λ is the current value of the Lagrange multiplier, then the new molecule becomes the new reference.

See also:
noprune::adjust_lagrange(), simple_prune::prune(), reorder_general_base::prune()

Here is the call graph for this function:



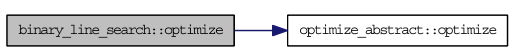
4.5.2.4 template<class C> ulong binary_line_search< C >::optimize (ulong N) const[inline, virtual]

This is a clean slate optimization. Any pruned molecules from previous runs are eradicated.

Parameters:
N gives the starting molecule's number in the library.

Implements optimize_abstract.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

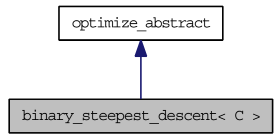
- binary_line_search.hh

4.6 binary_steepest_descent< C > Class Template Reference

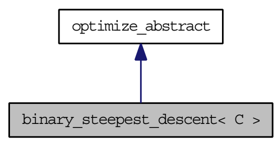
Searches a Library for an optimum using a steepest descent method.

#include <binarysteepestdescent.hh>

Inheritance diagram for binary_steepest_descent< C >:



Collaboration diagram for binary_steepest_descent< C >:



Public Member Functions

- ulong optimize (ulong N) const
  Optimize starting from index N.

Private Member Functions

- BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))
  Require C to conform to the Concepts::pruner concept.
- BOOST_CONCEPT_ASSERT ((Concepts::has_gradients< C >))
  Require C to conform to the Concepts::has_gradients concept.

4.6.1 Detailed Description

template<class C> class binary_steepest_descent< C >

As all optimization classes in the binary_* vein, the library is organized and searched as a hypercube on which the molecules are the vertices on the hull. The resultant bit-string representation of each molecule is used to perform optimizations.

4.6.2 Member Function Documentation

4.6.2.1 template<class C> binary_steepest_descent< C >::BOOST_CONCEPT_ASSERT ((Concepts::has_gradients< C >)) [private]

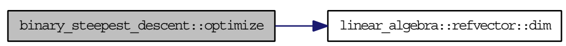
4.6.2.2 template<class C> binary_steepest_descent< C >::BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))[private]

4.6.2.3 template<class C> ulong binary_steepest_descent< C >::optimize (ulong N) const[inline, virtual]

Each edge (i.e., bit) on the hypercube is interpreted as a search direction. A gradient is computed along each direction and is then searched for improvements. Constraints are enforced with Lagrange multipliers which are ramped after a local optimum is reached.

Implements optimize_abstract.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

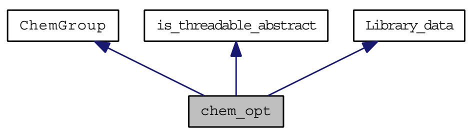
- binarysteepestdescent.hh

4.7 chem_opt Class Reference

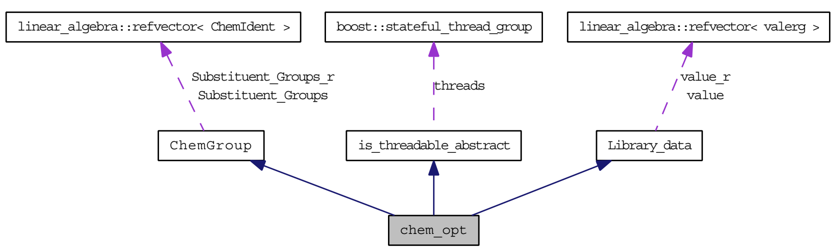
Chemical optimization class.

#include <chem_opt.hh>

Inheritance diagram for chem_opt:



Collaboration diagram for chem_opt:



Public Member Functions

- chem_opt ()
  Default constructor.
- chem_opt (const chem_opt &a)
  Default constructor.
- chem_opt (const ChemGroup &a)
  Construct from a ChemGroup.
- chem_opt (ulong nmax)
  Construction assigning maximum number of threads.

- chem_opt & operator= (const ChemGroup &a)
  Assignment operator.
- void output () const
  Output of the chemical pattern.
- valerg compute_property (ulong i) const
  Compute the property.
- ulong get_space_size () const
  Compute the size of the optimization space.
- ulong get_space_size (const long Group) const
  Compute the size of the optimization space of a specific group.
- ulong get_bits () const
  Compute the number of bits to address the optimization space.

4.7.1 Detailed Description

The class implements a Library of molecules that can be enumerated.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 chem_opt::chem_opt ()

4.7.2.2 chem_opt::chem_opt (const chem_opt & *a*)

4.7.2.3 chem_opt::chem_opt (const ChemGroup & *a*)

4.7.2.4 chem_opt::chem_opt (ulong *nmax*)

4.7.3 Member Function Documentation

4.7.3.1 valerg chem_opt::compute_property (ulong *i*) const[virtual]

Computes the Z-matrix of molecule i and then performs a conformational search and finally computes and returns the computed property value (as well as constraint violations).

Implements Library_data.

4.7.3.2 ulong chem_opt::get_bits () const[virtual]

Implements Library_data.

4.7.3.3 ulong chem_opt::get_space_size (const long *Group*) const

4.7.3.4 ulong chem_opt::get_space_size () const[virtual]

Implements Library_data.

4.7.3.5 chem_opt & chem_opt::operator= (const ChemGroup & *a*)

This should really never be invoked on anything that has already been started.

4.7.3.6 void chem_opt::output () const

Reimplemented from ChemGroup. The documentation for this class was generated from the following files:

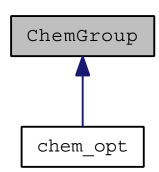
- chem_opt.hh
- chem_opt.cc

4.8 ChemGroup Class Reference

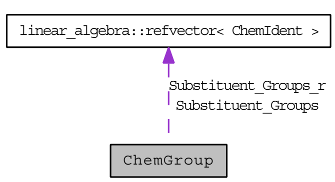
This class describes a group of substitution sites and their respective substitution options as well as computation history.

#include <chemgroup.hh>

Inheritance diagram for ChemGroup:



Collaboration diagram for ChemGroup:



Public Member Functions

- bool error_free () const
  Check whether there are any errors in the definition.
- mat build_zmat (long i, ulong &number, const zmat_connector &e, zmat &A, zmat_connector &y) const
  Build the Z-matrix using occupation i.
- zmat & build_zmat (long i, const zmat_connector &e, zmat &A, zmat_connector &y) const
  Build the Z-matrix using occupation i.
- void occupy (ulong number) const
  Set occupations according to number.
- ChemGroup & operator= (const ChemGroup &a)
  Assignment operator.
- bool operator== (const ChemGroup &a) const
  Comparison operator.
- void output () const
  Display to cout what's going on.

Constructors

- ChemGroup ()
  Empty constructor.
- ChemGroup (const ChemGroup &a)
  Copy constructor.
- ChemGroup (stringstream &in)
  Construction from a stringstream.
- ChemGroup (istream &in)
  Construction from a file.

Substituent manipulations

- long add_substituent (const ChemIdent &a)
  Add a substituent.

- refvector< long > add_substituents (const refvector< ChemIdent > &a)
  Add a list of substituents.
- void add_substituent (long i, long j, long k)
  Add a substituent to a site.
- void add_substituents (long i, long k, const refvecto< long > &j)
  Add a list of substituents to a site.

Public Attributes

- const refvector< ChemIdent > & Substituent_Groups_r

Private Attributes

- refvector< ChemIdent > Substituent_Groups

4.8.1 Detailed Description

It assumes the existence of a valerg calc_property(const zmat& A, const stringstream& id) function to compute properties. In essence this is a collection of ChemIdent objects. The first substitution group is used as the root for generating any Z-matrices based on the specific chosen substitutions.

The general strategy is to split each substitution group (ChemIdent) from the overall structure. Hence, a ChemIdent identifies a molecular framework/scaffold and possible substitution sites on this. Singletons don't have any substitution sites and cap substitutions. The ChemGroup collects these and gives meaning to specific substitutions at each defined site.

See class ChemIdent for the substitution rules. **Examples:**

carbazoles.inp, and vanilla-rings.inp.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 ChemGroup::ChemGroup ()

4.8.2.2 ChemGroup::ChemGroup (const ChemGroup & *a*)

4.8.2.3 ChemGroup::ChemGroup (stringstream & *s*)

See also:

ChemGroup::ChemGroup(istream& in)

4.8.2.4 ChemGroup::ChemGroup (istream & *in*)

The input file format is as follows:ChemGroup(ChemIdent1 ChemIdent2 ...)

See also:

ChemIdent::ChemIdent(istream& in) for ChemIdent input format and examples.

4.8.3 Member Function Documentation

4.8.3.1 void ChemGroup::add_substituent (long *i*, long *j*, long *k*)

A new substituent is added at site j of group i. k references the global group that is taken from ChemGroup and linked to the substition site. Hence, j has to be a valid site on Substituent_Groups[i] and k as well as i are an index of Substituent_Groups.

4.8.3.2 long ChemGroup::add_substituent (const ChemIdent & *a*)

This appends another ChemIdent object to the list of possible substution groups.

4.8.3.3 void ChemGroup::add_substituents (long *i*, long *m*, const refvecto< long > & *j*)

This adds a number of substituents to site m on group i. Similar to add_substituent().

4.8.3.4 refvector< long > ChemGroup::add_substituents (const refvector< ChemIdent > & *a*)

This appends a list of substituents as add_substituent() does for a single group.

4.8.3.5 zmat & ChemGroup::build_zmat (long *Group*, const zmat_connector & *e*, zmat & *A*, zmat_connector & *y*) const

This routine generates the local Z-matrix and combines it with the global Z-matrix using the default values as provided by occupation. Due to connectivity concerns it may be necessary to define appropriate dummy atoms in zmat::Z. A suggestion would be a dummy atom for each substitution group to define the topology.

Parameters:

Group index of the group to be added to the Z-matrix
e defines the connection of Group to the Z-matrix
A is the Z-matrix to which the Group topology will be attached. It will also be returned.

Note the non-const reference.
y defines the return connector.

4.8.3.6 zmat & ChemGroup::build_zmat (long *Group*, ulong & *number*, const zmat_connector & *e*, zmat & *A*, zmat_connector & *y*) const

This routine generates the local Z-matrix and combines it with the global Z-matrix for molecule number. Due to connectivity concerns it may be necessary to define appropriate dummy atoms in zmat::Z. A suggestion would be a dummy atom for each substitution group to define the topology. The combinatorics are different in this version, as it frees all constraints imposed by the general structure.

Parameters:

Group index of the group to be added to the Z-matrix
number index of the molecule to be built.
A is the Z-matrix to which the Group topology will be attached. It will also be returned. Note the non-const reference.
y defines the return connector.

4.8.3.7 bool ChemGroup::error_free () const

4.8.3.8 void ChemGroup::occupy (ulong *number*) const

Each ChemIdent has default occupations for its sites. This sets these to reflect the molecule referenced by number.

4.8.3.9 ChemGroup & ChemGroup::operator= (const ChemGroup & *a*)

Assignment operator.

Reimplemented in chem_opt.

4.8.3.10 bool ChemGroup::operator== (const ChemGroup & *a*) const

4.8.3.11 void ChemGroup::output () const

Reimplemented in chem_opt.

4.8.4 Member Data Documentation

4.8.4.1 refvector<ChemIdent> ChemGroup::Substituent_Groups [private]

This array holds the possible ChemIdent substitutions for each substitution site.

4.8.4.2 const refvector<ChemIdent>& ChemGroup::Substituent_Groups_r

This double array holds the possible substitutions for each substitution site. (READ ONLY)

The documentation for this class was generated from the following files:

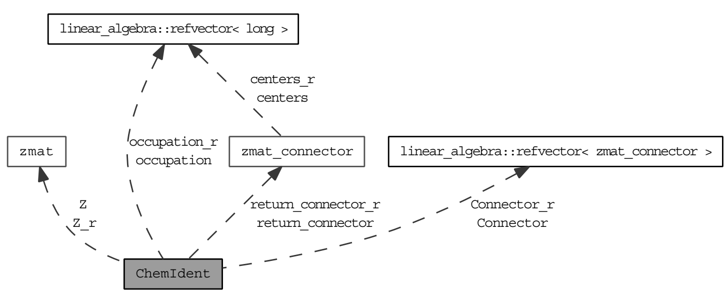
- chemgroup.h
- chemgroup.cc

4.9 ChemIdent Class Reference

This class describes a group of substitution sites on a Z-matrix and their respective substitution options.

#include <chemident.hh>

Collaboration diagram for ChemIdent:



Public Member Functions

- refvector< int > & generate_conformation_list (long i, const zmat_connector &e, refvector< int > &conformation, zmat_connector &y) const
  Build a list of indices for occupied substituents, configuration only.(DEPRECATED).
- ChemIdent & operator= (const ChemIdent &a)
  Assignment operator.
- bool operator== (const ChemIdent &a) const
  Comparison operator.

- ChemIdent & set_Z (const zmat &Z)
  Set the Z-matrix.
- ChemIdent & set_return_connector (const zmat_connector &A)
  Set the return connector.
- ChemIdent & add_to_dihedrals (int zentry, int maxn, int n)
  Add another dihedral for conformational searching.
- ChemIdent & set_dihedrals (int dihedrals, int n)
  Set the value of a dihedral.
- void occupy (long i, long j) const
  Set occupation to generate the zmat.
- long compute_space_size () const
  Computes the size of the library of molecules described by this instance.
- void fix_Substituent_dihedrals ()
  Fix the substituent dihedrals for connection purposes.
- void output () const
  Display to cout what's going on.

Constructors

- ChemIdent ()
  Empty constructor.
- ChemIdent (const string &Name)
  Primary constructor.
- ChemIdent (const zmat &A)
  Primary constructor from a Z-matrix.
- ChemIdent (const ChemIdent &a)
  Copy constructor.
- ChemIdent (stringstream &in)
  Construction from a stringstream.
- ChemIden (istream &in)
  Construction from a file.

Site manipulations

- ChemIdent & add_substitution_site (const refvector< long > &a, const zmat_connector &e)
  Increase the number of substitution sites and add a list of substituents for that site.

- ChemIdent & add_substitution_site (long a, const zmat_connector &e)
  Increase the number of substitution sites and add a substituent number to that site.

Substituent manipulations

- ChemIdent & add_substituents (long i, const refvector< long > &a)
  Add a substituent.

- ChemIdent & add_substituent (long i, long j)
  Add a substituent.

Static Public Member Functions

- static zmat_entry & update_connector (const zmat_connector &a, const zmat_connector &e, const long A, zmat_connector &x)
  Update a connector to fit, e.g., after combining two Z-matrices.

Public Attributes

- const refvector< refvector< long > > & allowed_Substituents_r

- const refvector< long > & occupation_r

- const zma & Z_

- const refvector< zmat_connector > & Connector_r

- const zmat_connector & return_connector_r

Private Attributes

- refvector< refvector< long > > allowed_
- refvector< long > occupation
- long Space_Size
- zmat Z
- refvector< zmat_connector > Connector
- zmat_connector return_connector

4.9.1 Detailed Description

The representation of a chemical substitution pattern is subdivided into a list of substitution sites which are potentially linked (return_connector). In the case of linkage the groups are considered to be linked in a chain, i.e., only to the predecessor. The return_connector has three distinct eferences:

- –6 to –4 refer to the original connector of the following group. (default)
- –3 to –1 refer to values of the connector passed down to the current group. Values greater then –1 refer to values in the local Z-matrix.

For the purpose of preoptimizing starting structures, which in the construction routine may be overlapped, the zmat_connector::opt_val and zmat_entry::opt_val structures direct the optimization of a specific entry. By default all dihedral angles between adjoined Z-matrices are optimized. This may cause problems in ring-structures. To change this the fix_Substituent_dihedrals() call cancels this behavior. Additionally it is possible to select dihedrals for conformational searching via ChemIdent::dihedrals. zmat_connector supplies zmat_connector::angle and zmat_connector::angle_val for conformational optimizations with respect to the connectors.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 ChemIdent::ChemIdent ()

4.9.2.2 ChIdent::ChIdent (const string & *Name*)

4.9.2.3 ChemIdent::ChemIdent (const zmat & *A*)

4.9.2.4 ChemIdent::ChemIdent (const ChemIdent & *a*)

4.9.2.5 ChemIdent::ChemIdent (stringstream & *s*)

The input file format is as follows:(Z(zmat)

ReturnConnector (zmat_connector)

Connector (zmat_connector zmat_connector ...)

allowed_Substituents((group number 1, group number 2 , ...) ...))

Under Z a zmat object is read in. The appropriate format can be found under zmat::zmat(stringstream& s). ReturnConnector reads in a zmat_connector using zmat_connector::zmat_connector(stringstream& s). Connector reads in a list of zmat_connector objects, each adhering to the format found under zmat_connector::zmat_connector(stringstream& s).

See also:
carbazoles.inp and vanilla-rings.inp in the examples section.

Read the connector.

Read the connector.

4.9.2.6 ChemIdent::ChemIdent (istream & *in*)

See also:
ChemIdent::ChemIdent(stringstream& in)

4.9.3 Member Function Documentation

4.9.3.1 ChemIdent & ChemIdent::add_substituent (long *i*, long *j*)

4.9.3.2 ChemIdent & ChemIdent::add_substituents (long *i*, const refvector< long > & *a*)

4.9.3.3 ChemIdent & ChemIdent::add_substitution_site (long *a*, const zmat_connector & *e*)

4.9.3.4 ChemIdent & ChemIdent::add_substitution_site (const refvector< long > & *a*, const zmat_connector & *e*)

4.9.3.5 ChemIdent & ChemIdent::add_to_dihedrals (int *zentry*, int *maxn*, int *n*)

Parameters:

zentry dihedral of Z-matrix entry to be optimized.

$maxn$ is the number of division of 360 deg, $\Delta\angle : \frac{360\,\mathrm{deg}}{maxn}$.
n current instantiation (<maxn).

4.9.3.6 long ChemIdent::compute_space_size () const

4.9.3.7 void ChemIdent::fix_Substituent_dihedrals ()

4.9.3.8 refvector<int>& ChemIdent::generate_conformation_list (long i, const zmat_connector & e, refvector< int > & *conformation*, zmat_connector & y) const

4.9.3.9 void ChemIdent::occupy (long i, long j) const

4.9.3.10 ChemIdent & ChemIdent::operator= (const ChemIdent & a)

Assignment operator.

4.9.3.11 bool ChemIdent::operator== (const ChemIdent & a) const

Comparison operator.

4.9.3.12 void ChemIdent::output () const

4.9.3.13 ChemIdent& ChemIdent::set_dihedrals (int *dihedrals*, int n)

4.9.3.14 ChemIdent & ChemIdent::set_return_connector (const zmat_connector & A)

4.9.3.15 ChemIdent & ChemIdent::set_Z (const zmat & Z)

4.9.3.16 zmat_entry& ChemIdent::update_connector (const zmat_connector & a, const zmat_connector & e, const long A, zmat_connector & x)[static]

4.9.4 Member Data Documentation

4.9.4.1 refvector<refvector<long> > ChemIdent::allowed_Substituents[private]

This double array holds the possible substitutions for each substitution site.

4.9.4.2 const refvector<refvector<long> >& ChemIdent::allowed_Substituents_r

This double array holds the possible substitutions for each substitution site. (READ ONLY)

4.9.4.3 refvector<zmat_connector> ChemIdent::Connector[private]

This array of entries holds the connectivity data on the substituents within Z and the

complete framework.

4.9.4.4 refvector<zmat_connector>& ChemIdent::Connector_r

This array of entries holds the connectivity data on the substituents within Z and the complete framework. (READ ONLY)

4.9.4.5 refvector<long> ChemIdent::occupation[mutable, private]

This array holds current substituents at each site

4.9.4.6 refvecto<long>ChIdent::occupation_r

This array holds current substituents at each site. (READ ONLY)

4.9.4.7 zmat_connector ChemIdent::return_connector[private]

This Z-matrix entry is returned in order to connect subsequent groups to this chemgroup.

4.9.4.8 const zmat_connector& ChemIdent::return_connector_r

This Z-matrix entry is returned in order to connect subsequent groups to this chemgroup.

4.9.4.9 long ChemIdent::Space_Size[mutable, private]

This number holds the possible number of combinations for the substitution sites.

4.9.4.10 zmat ChemIdent::Z[private]

This Z-matrix holds the connectivity data for this group. Respective substitution sites will be added via their own build_zmat() calls. For unambiguous building of Z-matrices it may be necessary to include dummy atoms. These are easily omitted upon conversion to cartesian coordinates.

4.9.4.11 const zma& ChemIdent::Z_r

This Z-matrix holds the connectivity data for this group. Respective substitution sites will be added via their own build_zmat() calls. For unambiguous building of Z-matrices it may be necessary to include dummy atoms. These are easily omitted upon conversion to cartesian coordinates. (READ ONLY)

The documentation for this class was generated from the following files:

- chemident.h

- chemident.cc

4.10 double_truct Reference

Pair of configurational and conformational index, respectively.

#include <typedefs.hh>

Public Attributes

- int configuration
- int conformation

4.10.1 Member Data Documentation

4.10.1.1 int double_index::configuration

4.10.1.2 int double_index::conformation

The documentation for this struct was generated from the following file:

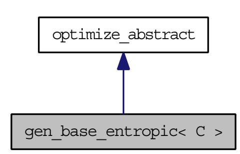
- typedefs.hh

4.11 gen_base_entropic< C > Class Template Reference

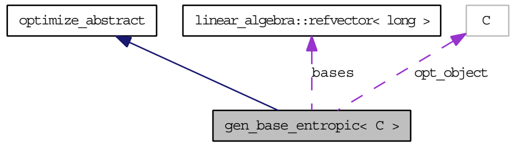
Meta-Optimize by generating maximally distant starting configurations.

#include <gen_base_entropic.hh>

Inheritance diagram for gen_base_entropic< C >:



Collaboration diagram for gen_base_entropic< C >:



Public Member Functions

- gen_base_entropic (const C &a, const refvector< long > &b)
- ulong optimize (ulong N) const
  Meta-Optimize by generating maximally distant starting configurations.
- valerg get_value (const ulong i) const

Public Attributes

- ulong nruns
  Number of runs.
- long max_steps
  Maximum number of steps.

Protected Attributes

- C opt_object
- refvector< long > bases

Private Member Functions

- BOOST_CONCEPT_ASSERT ((Concepts::has_optimize< C >))
  Require C to conform to an optimization.
- BOOST_CONCEPT_ASSERT ((boost::CopyConstructible< C >))
  Require Copy constructibility.

template<class C> class gen_base_entropic< C >

4.11.1 Constructor & Destructor Documentation

4.11.1.1 template<class C> gen_base_entropic< C >::gen_base_entropic (const C & *a*, const refvector< long > & *b*)[inline]

4.11.2 Member Function Documentation

4.11.2.1 template<class C> gen_base_entropic< C >::BOOST_CONCEPT_−ASSERT ((boost::CopyConstructible< C >))[private]

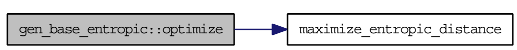
4.11.2.2 template<class C> gen_base_entropic< C >::BOOST_CONCEPT_ASSERT ((Concepts::has_optimize< C >))[private]

4.11.2.3 template<class C> valerg gen_base_entropic< C >::get_value (const ulong *i*) const[inline]

4.11.2.4 template<class C> ulong gen_base_entropic< C >::optimize (ulong *N*) const[inline, virtual]

Implements optimize_abstract.

Here is the call graph for this function:



4.11.3 Member Data Documentation

4.11.3.1 template<class C> refvector<long> gen_base_entropic< C >::bases[protected]

4.11.3.2 template<class C> long gen_base_entropic< C >::max_steps

4.11.3.3 template<class C> ulong gen_base_entropic< C >::nruns

4.11.3.4 template<class C> C gen_base_entropic< C >::opt_object[protected]

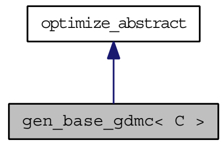
The documentation for this class was generated from the following file:

- gen_base_entropic.hh

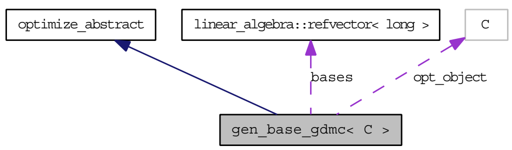
4.12 gen_base_gdmc< C > Class Template Reference

#include <genbasegdmc.hh>

Inheritance diagram for gen_base_gdmc< C >:



Collaboration diagram for gen_base_gdmc< C >:



Public Member Functions

- gen_base_gdmc (const C &a, const refvector< long > &b)
- ulong optimize (ulong N) const
  Gradient-directed Monte-Carlo optimization.
- valerg get_value (ulong i) const

Public Attributes

- double T
  Temperature.
- ulong tight_steps
  Number of tightening steps.
- ulong max_steps
  Maximum number of steps.

Private Member Functions

- BOOST_CONCEPT_ASSERT ((Concepts::has_gradients< C >))
  Require C to conform to the has_gradients concept.
- BOOST_CONCEPT_ASSERT ((Concepts::has_optimize< C >))
  Require C to conform to an optimization.
- BOOST_CONCEPT_ASSERT ((Concepts::has_stacksize< C >))
  Require C to conform to providing computed number of configurations.
- BOOST_CONCEPT_ASSERT ((boost::CopyConstructible< C >))
  Require Copy constructibility.

Private Attributes

- C opt_object
- refvector< long > bases

template<class C> class gen_base_gdmc< C >

4.12.1 Constructor & Destructor Documentation

4.12.1.1 template<class C> gen_base_gdmc< C >::gen_base_gdmc (const C & *a*, const refvector< long > & *b*)[inline]

4.12.2 Member Function Documentation

4.12.2.1 template<class C> gen_base_gdmc< C >::BOOST_CONCEPT_ASSERT ((boost::CopyConstructible< C >))[private]

4.12.2.2 template<class C> gen_base_gdmc< C >::BOOST_CONCEPT_ASSERT ((Concepts::has_stacksize< C >))[private]

4.12.2.3 template<class C> gen_base_gdmc< C >::BOOST_CONCEPT_ASSERT ((Concepts::has_optimize< C >))[private]

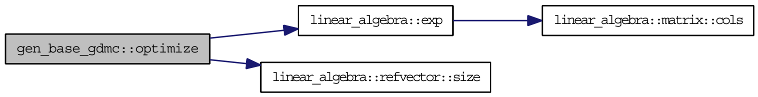
4.12.2.4 template<class C> gen_base_gdmc< C >::BOOST_CONCEPT_ASSERT ((Concepts::has_gradients< C >))[private]

4.12.2.5 template<class C> valerg gen_base_gdmc< C >::get_value (ulong *i*) const[inline]

4.12.2.6 template<class C> ulong gen_base_gdmc< C >::optimize (ulong *N*) const[inline, virtual]

Implements optimize_abstract.

Here is the call graph for this function:



4.12.3 Member Data Documentation

4.12.3.1 template<class C> refvector<long> gen_base_gdmc< C >::bases[private]

4.12.3.2 template<class C> ulong gen_base_gdmc< C >::max_steps

4.12.3.3 template<class C> C gen_base_gdmc< C >::opt_object[private]

4.12.3.4 template<class C> double gen_base_gdmc< C >::T

4.12.3.5 template<class C> ulong gen_base_gdmc< C >::tight_steps

The documentation for this class was generated from the following file:

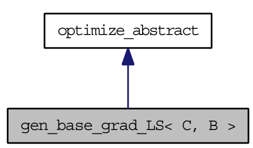
- genbasegdmc.hh

4.13 gen_base_grad_LS< C, B > Class Template Reference

Line search optimization class using general bases.

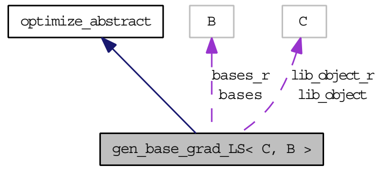
#include <genbase-grad-ls.hh>

Inheritance diagram for gen_base_grad_LS< C, B >:



Collaboration diagram for gen_base_grad_LS< C, B >:

Public Member Functions



- gen_base_grad_LS (const C &Library)
- gen_base_grad_LS (const gen_base_grad_LS< C, B > &a)
  Copy constructor.
- refvector< valerg > gradient (const ulong conf1) const
  Gradient computation.
- refvector< valerg > & gradient (const ulong conf1, refvector< valerg > &r) const
  Gradient computation.
- long stacksize () const
- ulong optimize (ulong N) const
  Optimize starting from conformation i.
- valerg get_value (const ulong i) const

Public Attributes

- const B & bases_r
- const C & lib_object_r

Private Member Functions

- BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))
  Require Concepts::pruner compliance of C.
- BOOST_CONCEPT_ASSERT ((boost::CopyConstructible< C >))
- BOOST_CONCEPT_ASSERT ((Concepts::base_iterator< B >))

Private Attributes

- const C lib_object
- B bases

template<class C, class B> class gen_base_grad_LS< C, B >

4.13.1 Constructor & Destructor Documentation

4.13.1.1 template<class C, class B> gen_base_grad_LS< C, B >::gen_base_grad_LS (const C & *Library*)[inline]

4.13.1.2 template<class C, class B> gen_base_grad_LS< C, B >::gen_base_grad_LS (const gen_base_grad_LS< C, B > & *a*)[inline]

4.13.2 Member Function Documentation

4.13.2.1 template<class C, class B> gen_base_grad_LS< C, B >::BOOST_CONCEPT_ASSERT ((Concepts::base_iterator< B >))[private]

4.13.2.2 template<class C, class B> gen_base_grad_LS< C, B >::BOOST_CONCEPT_ASSERT ((boost::CopyConstructible< C >))[private]

4.13.2.3 template<class C, class B> gen_base_grad_LS< C, B >::BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))[private]

4.13.2.4 template<class C, class B> valerg gen_base_grad_LS< C, B >::get_value (const ulong *i*) const[inline]

4.13.2.5 template<class C, class B> refvector<valerg>& gen_base_grad_LS< C, B >::gradient (const ulong *conf1*, refvecto< valerg > & *r*) const[inline]

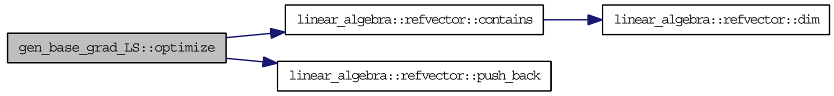
4.13.2.6 template<class C, class B> refvector<valerg> gen_base_grad_LS< C, B >::gradient (const ulong *conf1*) const[inline]

4.13.2.7 template<class C, class B> ulong gen_base_grad_LS< C, B >::optimize (ulong *N*) const[inline, virtual]

A Lagrange multiplier method is employed to enforce boundary constraints. The multiplier increases as the optimizatoin proceeds, enforcing the constraint ever more rigorously.

Implements optimize_abstract.

Here is the call graph for this function:



4.13.2.8 template<class C, class B> long gen_base_grad_LS< C, B >::stacksize () const[inline]

4.13.3 Member Data Documentation

4.13.3.1 template<class C, class B> B gen_base_grad_LS< C, B >::[mutable, private]

4.13.3.2 template<class C, class B> const B& gen_base_grad_LS< C, B >::bases_r

4.13.3.3 template<class C, class B> const C gen_base_grad_LS< C, B >::lib_object[private]

4.13.3.4 template<class C, class B> const C& gen_base_grad_LS< C, B >::lib_object_r

The documentation for this class was generated from the following file:

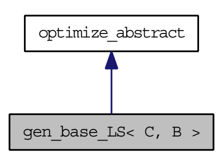
- genbase-grad-ls.hh

4.14 gen_base_LS< C, B > Class Template Reference

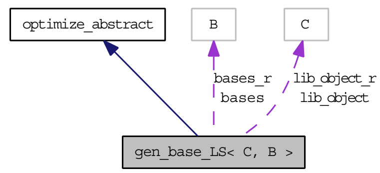
Line search optimization class using general bases.

#include <genbase-l-s.hh>

Inheritance diagram for gen_base_LS< C, B >:



Collaboration diagram for gen_base_LS< C, B >:



*Public Member Functions

- gen_base_LS (const C &Library)
- ulong optimize (ulong N) const
  Optimize starting from conformation i.
- valerg get_value (const ulong i) const

Public Attributes

- const B bases_r
- const C & lib_object_r

Private Member Functions

- BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))
  Require Concepts::pruner compliance of C.
- BOOST_CONCEPT_ASSERT ((boost::CopyConstructible< C >))
- BOOST_CONCEPT_ASSERT ((Concepts::base_iterator< B >))

Private Attributes

- const C lib_object
- B bases

template<class C, class B> class gen_base_LS< C, B >

4.14.1 Constructor & Destructor Documentation

4.14.1.1 template<class C, class B> gen_base_LS< C, B >::gen_base_LS (const C & *Library*)[inline]

4.14.2 Member Function Documentation

4.14.2.1 template<class C, class B> gen_base_LS< C, B >::BOOST_CONCEPT_ASSERT ((Concepts::base_iterator< B >))`[private]`

4.14.2.2 template<class C, class B> gen_base_LS< C, B >::BOOST_CONCEPT_ASSERT ((boost::CopyConstructible< C >))[private]

4.14.2.3 template<class C, class B> gen_base_LS< C, B >::BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))[private]

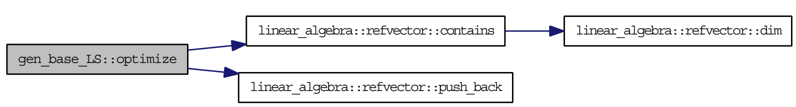
4.14.2.4 template<class C, class B> valerg gen_base_LS< C, B >::get_value (const ulong *i*) const[inline]

4.14.2.5 template<class C, class B> ulong gen_base_LS< C, B >::optimize (ulong *N*) const[inline, virtual]

A Lagrange multiplier method is employed to enforce boundary constraints. The multiplier increases as the optimizatoin proceeds, enforcing the constraint ever more rigorously.

Implements optimize_abstract.

Here is the call graph for this function:



4.14.3 Member Data Documentation

4.14.3.1 template<class C, class B> B gen_base_LS< C, B >::bases[mutable, private]

4.14.3.2 template<class C, class B> const B& gen_base_LS< C, B >::bases_r

4.14.3.3 template<class C, class B> const C gen_base_LS< C, B >::lib_object[private]

4.14.3.4 template<class C, class B> const C& gen_base_LS< C, B >::lib_object_r

The documentation for this class was generated from the following file:

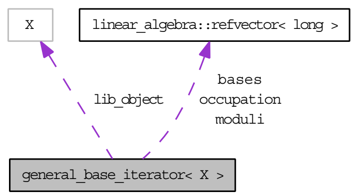
- genbase-l-s.hh

4.15 general_base_iterator< X > Class Template Reference

Iterator over the potential bases.

#include <generalbaseiterator.hh>

Collaboration diagram for general_base_iterator< X >:



Public Member Functions

- general_base_iterator (const X &b)
  Constructor. Mandatory library object.
- general_base_iterator (const general_base_iterator< X > &b)
  Copy Constructor.
- general_base_iterator< X > & operator++ (int i)
  Advance the iterator to the next number.
- ulong set_refstate (ulong newref)
  Set the reference state of the library to determine the relevance of subs.
- ulong get_refstate () const
  Retrieve the current refstate.
- long operator= (long i)
  Set the current state.
- long get_state () const
  Retrieve the current state.

- long operator() () const
  Retrieve the current base size.
- long modulus () const
  Modulus.
- bool done () const
  End of iterator?
- long non_empty_size () const
- template<> bool done () const
- template<> void occupy (ulong number)
- template<> void compute_bases (long Group, const refvector< long > &base_sizes)
- template<> void compute_bases ()
- template<> general_base_iterator (const general_base_iterator< chem_opt > &b)
- template<> general_base_iterator (const chem_opt &b)
- template<> long modulus () const
- template<> general_base_iterator< chem_opt > & operator++ (int i)
- template<> ulong get_refstate () const
- template<>ulong set_refstate (ulong newref)
- template<> long operator= (long i)
- template<> long get_state () const
- template<> long operator() () const
- template<> long non_empty_size () const

Private Member Functions

- BOOST_CONCEPT_ASSERT ((Concepts::Library< X >))
- void compute_bases ()
  Compute offsets of groups.

- void compute_bases (long N, const refvector< long > &base_sizes)
  Compute the base for N.

- void occupy (ulong number)
  Set the occupation numbers w.r.t. number.

Private Attributes

- long number_of_bases
- ulong refstate
  Reference state.
- long state
  Current base.
- const X & lib_object
  Library object which is enumerated with the bases to be iterated.
- refvector< long > bases
  Offsets of the bases.
- refvector< long > moduli
  Moduli of the bases.
- refvector< long > occupation
  occupation of groups by current refstate.

template<class X> class general_base_iterator< X >

4.15.1 Constructor & Destructor Documentation

4.15.1.1 template<class X> general_base_iterator< X >::general_base_iterator (const X & *b*)

4.15.1.2 template<class X> general_base_iterator< X >::general_base_iterator (const general_base_iterator< X > & *b*)

4.15.1.3 template<> general_base_iterator< chem_opt >::general_base_iterator (const general_base_iterator< chem_opt > & *b*)

4.15.1.4 template<> general_base_iterator< chem_opt >::general_base_iterator (const chem_opt & *b*)

4.15.2 Member Function Documentation

4.15.2.1 template<class X> general_base_iterator< X >::BOOST_CONCEPT_ASSERT ((Concepts::Library< X >))[private]

4.15.2.2 template<> void general_base_iterator< chem_opt >::compute_bases ()

4.15.2.3 template<> void general_base_iterator< chem_opt >::compute_bases (long *Group*, const refvector< long > & *base_sizes*)

4.15.2.4 template<class X> void general_base_iterator< X >::compute_bases (long *N*, const refvector< long > & *base_sizes*)[private]

4.15.2.5 template<class X> void general_base_iterator< X >::compute_bases ()[private]

4.15.2.6 template<> bool general_base_iterator< chem_opt >::done () const

4.15.2.7 template<class X> bool general_base_iterator< X >::done () const

4.15.2.8 template<> ulong general_base_iterator< chem_opt >::get_refstate () const

4.15.2.9 template<class X> ulong general_base_iterator< X >::get_refstate () const

4.15.2.10 template<> long general_base_iterator< chem_opt >::get_state () const

4.15.2.11 template<class X> long general_base_iterator< X >::get_state () const

4.15.2.12 template<> long general_base_iterator< chem_opt >::modulus () const

4.15.2.13 template<class X> long general_base_iterator< X >::modulus () const

4.15.2.14 template<> long general_base_iterator< chem_opt >::non_empty_size () const

4.15.2.15 template<class X> long general_base_iterator< X >::non_empty_size () const

4.15.2.16 template<> void general_base_iterator< chem_opt >::occupy (ulong *number*)

4.15.2.17 template<class X> void general_base_iterator< X >::occupy (ulong *number*)[private]

4.15.2.18 template<> long general_base_iterator< chem_opt >::operator() () const

4.15.2.19 template<class X> long general_base_iterator< X >::operator() () const

4.15.2.20 template<> general_base_iterator< chem_opt > & general_base_iterator<

chem_opt >::operator++ (int *i*)

4.15.2.21 template<class X> general_base_iterator<X>& general_base_iterator< X >::operator++ (int *i*)

4.15.2.22 template<> long general_base_iterator< chem_opt >::operator= (long *i*)

4.15.2.23 template<class X> long general_base_iterator< X >::operator= (long *i*)

4.15.2.24 template<> ulong general_base_iterator< chem_opt >::set_refstate (ulong *newref*)

4.15.2.25 template<class X> ulong general_base_iterator< X >::set_refstate (ulong *newref*)

4.15.3 Member Data Documentation

4.15.3.1 template<class X> refvector<long> general_base_iterator< X >::bases[mutable, private]

4.15.3.2 template<class X> const X& general_base_iterator< X >::lib_object[private]

4.15.3.3 template<class X> refvector<long> general_base_iterator< X >::moduli[mutable, private]

4.15.3.4 template<class X> long general_base_iterator< X >::number_of_bases[private]

4.15.3.5 template<class X> refvector<long> general_base_iterator< X >::occupation[private]

4.15.3.6 template<class X> ulong general_base_iterator< X >::refstate[private]

4.15.3.7 template<class X> long general_base_iterator< X >::state[private]

The documentation for this class was generated from the following file:

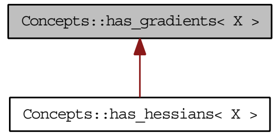
- generalbaseiterator.hh

4.16 Concepts::has_gradients< X > Class Template Reference

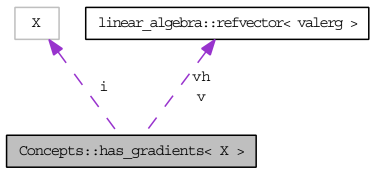
Concept defines a class that has gradients.

#include <concepts.hh>

Inheritance diagram for Concepts::has_gradients< X >:



Collaboration diagram for Concepts::has_gradients< X >:



Public Member Functions

- BOOST_CONCEPT_USAGE (has_gradients)

Private Attributes

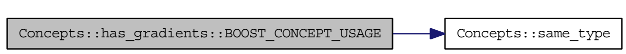
- X & i
- valer > v
- refvector< valerg > & vh

template<class X> class Concepts::has_gradients< X >

4.16.1 Member Function Documentation

4.16.1.1 template<class X> Concepts::has_gradients< X >::BOOST_CONCEPT_USAGE (has_gradients< X >)[inline]

Here is the call graph for this function:



4.16.2 Member Data Documentation

4.16.2.1 template<class X> X& Concepts::has_gradients< X >::i[private]

Reimplemented in Concepts::has_hessians< X >.

4.16.2.2 template<class X> refvector<valerg> Concepts::has_gradients< X >::v[private]

Reimplemented in Concepts::has_hessians< X >.

4.16.2.3 template<class X> refvector<valerg>& Concepts::has_gradients< X >::vh[private]

Reimplemented in Concepts::has_hessians< X >.

The documentation for this class was generated from the following file:

- concepts.hh

4.17 has_gradients_data< X > Class Template Reference

Abstract class for discrete optimizations.

Public Member Functions

- has_gradients_data ()
- has_gradients_data (const has_gradients_data &a)
- has_gradients_data & operator= (const has_gradients_data &d)
- void diff (ulong n1, ulong n2, valerg &r) const
  compute the difference between numbers n1 and n2.
- refvector< valerg > gradient (const ulong i) const
  Gradient computation.
- refvector< valerg > & gradient (const ulong i, refvector< valerg > &v) const
  Gradient computation.

Protected Member Functions

- void gradient_thread (boost::shared_mutex *mx, const ulong *i, ulong *k) const

Private Member Functions

- BOOST_CONCEPT_ASSERT ((Concepts::Library< X >))
- BOOST_CONCEPT_ASSERT ((Concepts::has_Name< X >))
- BOOST_CONCEPT_ASSERT ((Concepts::is_threadable< X >))

template<class X> class has_gradients_data< X >

4.17.1 Constructor & Destructor Documentation

4.17.1.1 template<class X> has_gradients_data< X >::has_gradients_data ()[inline]

4.17.1.2 template<class X> has_gradients_data< X >::has_gradients_data (const has_gradients_data< X > & *a*)[inline]

4.17.2 Member Function Documentation

4.17.2.1 template<class X> has_gradients_data< X >::BOOST_CONCEPT_ASSERT ((Concepts::is_threadable< X >))[private]

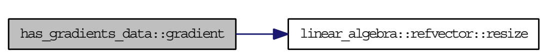
4.17.2.2 template<class X> has_gradients_data< X >::BOOST_CONCEPT_ASSERT ((Concepts::has_Name< X >))[private]

4.17.2.3 template<class X> has_gradients_data< X >::BOOST_CONCEPT_ASSERT ((Concepts::Library< X >))[private]

4.17.2.4 template<class X> void has_gradients_data< X >::diff (ulong *n1*, ulong *n2*, valerg & *r*) const

4.17.2.5 template<class X> refvector< valerg > & has_gradients_data< X >::gradient (const ulong *i*, refvector< valerg > & *v*) const

Here is the call graph for this function:



4.17.2.6 template<class X> refvector< valerg > has_gradients_data< X >::gradient (const ulong *i*) const

4.17.2.7 template<class X> void has_gradients_data< X >::gradient_thread (boost::shared_mutex * mx, const ulong * i, ulong * k) const[protected]

4.17.2.8 template<class X> has_gradients_data& has_gradients_data< X >::operator= (const has_gradients_data< X > & d)

The documentation for this class was generated from the following files:

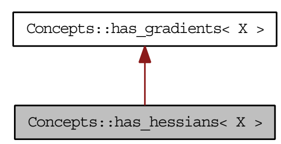
- has_gradients_hessian_data.decl
- has_gradients_hessian_data.hh

4.18 Concepts::has_hessians< X > Class Template Reference

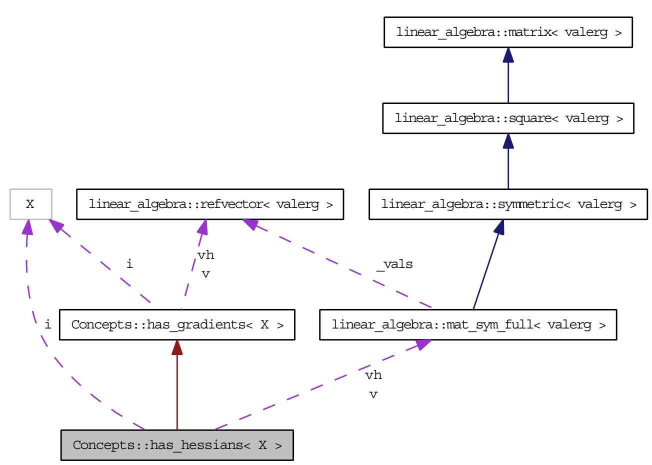
Concept defines a class that has hessians.

#include <concepts.hh>

Inheritance diagram for Concepts::has_hessians< X >:



Collaboration diagram for Concepts::has_hessians< X >:



Public Member Functions

- BOOST_CONCEPT_USAGE (has_hessians)

- X i
- mat_sym_full< valerg > v
- mat_sym_full< valerg > & vh

template<class X> class Concepts::has_hessians< X >

4.18.1 Member Function Documentation

4.18.1.1 template<class X> Concepts::has_hessians< X >::BOOST_CONCEPT_USAGE (has_hessians< X >)[inline]

Here is the call graph for this function:

4.18.2 Member Data Documentation

4.18.2.1 template<class X> X Concepts::has_hessians< X >::i[private]

Reimplemented from Concepts::has_gradients< X >.

4.18.2.2 template<class X> mat_sym_full<valerg> Concepts::has_hessians< X >::v[private]

Reimplemented from Concepts::has_gradients< X >.

4.18.2.3 template<class X> mat_sym_full<valerg>& Concepts::has_hessians< X >::vh[private]

Reimplemented from Concepts::has_gradients< X >.

The documentation for this class was generated from the following file:

- concepts.hh

4.19 has_hessians_data< X > Class Template Reference

Abstract class for discrete optimizations.

Public Member Functions

- has_hessians_data ()
- has_hessians_data (const has_hessians_data &a)
- has_hessians_data & operator= (const has_hessians_data &d)

- mat_sym_full< valerg > hessian (ulong i) const
  Hessian computation.
- mat_sym_full< valerg > & hessian (ulong i, mat_sym_full< valerg > &H) const
  Hessian computation.

Public Attributes

- const string & Name_r

Private Member Functions

- BOOST_CONCEPT_ASSERT ((Concepts::has_gradients< X >))
- BOOST_CONCEPT_ASSERT ((Concepts::Library< X >))
- BOOST_CONCEPT_ASSERT ((Concepts::has_Name< X >))
- BOOST_CONCEPT_ASSERT ((Concepts::is_threadable< X >))

template<class X> class has_hessians_data< X >

4.19.1 Constructor & Destructor Documentation

4.19.1.1 template<class X> has_hessians_data< X >::has_hessians_data ()[inline]

4.19.1.2 template<class X> has_hessians_data< X >::has_hessians_data (const has_hessians_data< X > & *a*)[inline]

4.19.2 Member Function Documentation

4.19.2.1 template<class X> has_hessians_data< X >::BOOST_CONCEPT_ASSERT ((Concepts::is_threadable< X >))[private]

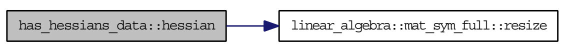
4.19.2.2 template<class X> has_hessians_data< X >::BOOST_CONCEPT_ASSERT ((Concepts::has_Name< X >))[private]

4.19.2.3 template<class X> has_hessians_data< X >::BOOST_CONCEPT_ASSERT ((Concepts::Library< X >))[private]

4.19.2.4 template<class X> has_hessians_data< X >::BOOST_CONCEPT_ASSERT ((Concepts::has_gradients< X >))[private]

4.19.2.5 template<class X> mat_sym_full< valerg > & has_hessians_data< X >::hessian (ulong i, mat_sym_full< valerg > & H) const

Here is the call graph for this function:



4.19.2.6 template<class X> mat_sym_full< valerg > has_hessians_data< X >::hessian (ulong i) const

4.19.2.7 template<class X> has_hessians_data& has_hessians_data< X >::operator= (const has_hessians_data< X > & d)

4.19.3 Member Data Documentation

4.19.3.1 template<class X> const string& has_hessians_data< X >::Name_r

The documentation for this class was generated from the following files:

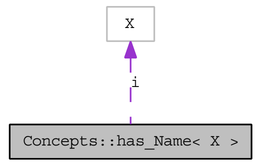
- has_gradients_hessian_data.decl
- has_gradients_hessian_data.hh

4.20 Concepts::has_Name< X > Class Template Reference

Concept defines a class that has a private string Name.

#include <concepts.hh>

Collaboration diagram for Concepts::has_Name< X >:



Public Member Functions

- BOOST_CONCEPT_USAGE (has_Name)

Private Attributes

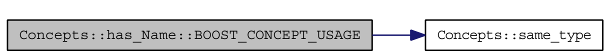
- X & i
- string N
- const string & Nh
- string M

template<class X> class Concepts::has_Name< X >

4.20.1 Member Function Documentation

4.20.1.1 template<class X> Concepts::has_Name< X >::BOOST_CONCEPT_USAGE (has_Name< X >)[inline]

Here is the call graph for this function:



4.20.2 Member Data Documentation

4.20.2.1 template<class X> X& Concepts::has_Name< X >::i[private]

4.20.2.2 template<class X> string Concepts::has_Name< X >::M[private]

4.20.2.3 template<class X> string Concepts::has_Name< X >::N[mutable, private]

4.20.2.4 template<class X> const string& Concepts::has_Name< X >::Nh[private]

The documentation for this class was generated from the following file:

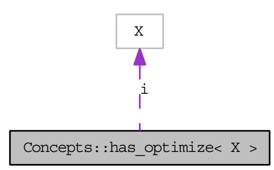
- concepts.hh

4.21 Concepts::has_optimize< X > Class Template Reference

Concept defines a class that has an optimize member.

#include <concepts.hh>

Collaboration diagram for Concepts::has_optimize< X >:



Public Member Functions

- BOOST_CONCEPT_USAGE (has_optimize)

Private Attributes

- X & i
- ulong index
- string & id

template<class X> class Concepts::has_optimize< X >

4.21.1 Member Function Documentation

4.21.1.1 template<class X> Concepts::has_optimize< X >::BOOST_CONCEPT_USAGE (has_optimize< X >)[inline]

Here is the call graph for this function:

4.21.2 Member Data Documentation

4.21.2.1 template<class X> X& Concepts::has_optimize< X >::*i*[private]

4.21.2.2 template<class X> string& Concepts::has_optimize< X >::id[private]

4.21.2.3 template<class X> ulong Concepts::has_optimize< X >::index[private]

The documentation for this class was generated from the following file:

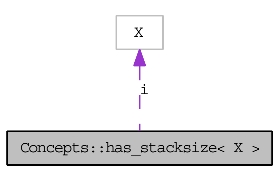
- concepts.hh

4.22 Concepts::has_stacksize< X > Class Template Reference

Concept defines a class that has a stack member.

#include <concepts.hh>

Collaboration diagram for Concepts::has_stacksize< X >:



Public Member Functions

- BOOST_CONCEPT_USAGE (has_stacksize)

Private Attributes

- X & i
- long index
- string & id

template<class X> class Concepts::has_stacksize< X >

4.22.1 Member Function Documentation

4.22.1.1 template<class X> Concepts::has_stacksize< X >::BOOST_CONCEPT_USAGE (has_stacksize< X >)[inline]

Here is the call graph for this function:

4.22.2 Member Data Documentation

4.22.2.1 template<class X> X& Concepts::has_stacksize< X >::i[private]

4.22.2.2 template<class X> string& Concepts::has_stacksize< X >::id[private]

4.22.2.3 template<class X> long Concepts::has_stacksize< X >::index[private]

The documentation for this class was generated from the following file:

- concepts.hh

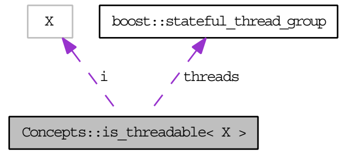
4.23 Concepts::is_threadable< X > Class Template Reference

Concept defines a class that is threadable.

#include <concepts.hh>

Collaboration diagram for Concepts::is_threadable< X >:

Public Member Functions



- BOOST_CONCEPT_USAGE (is_threadable)

Private Attributes

- X i
- ulong mt
- boost::shared_mutex hm
- boost::stateful_thread_group threads

template<class X> class Concepts::is_threadable< X >

4.23.1 Member Function Documentation

4.23.1.1 template<class X> Concepts::is_threadable< X >::BOOST_CONCEPT_USAGE (is_threadable< X >)[inline]

Here is the call graph for this function:

4.23.2 Member Data Documentation

4.23.2.1 template<class X> boost::shared_mutex Concepts::is_threadabl< X >::hm[mutable, private]

4.23.2.2 template<class X> X Concepts::is_threadable< X >::i[private]

4.23.2.3 template<class X> ulong Concepts::is_threadable< X >::mt[private]

4.23.2.4 template<class X> boost::stateful_thread_group Concepts::is_threadable< X >::threads[mutable, private]

The documentation for this class was generated from the following file:

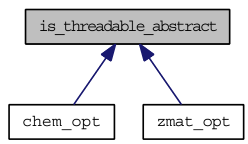
- concepts.hh

4.24 is_threadable_abstract Class Reference

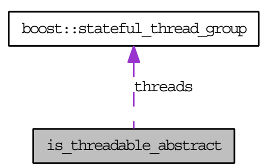
Abstract class implementing the Concepts::is_threadable concept.

#include <isthreadableabstract.h>

Inheritance diagram for is_threadable_abstract:



Collaboration diagram for is_threadable_abstract:



Public Member Functions

- void set_max_threads (ulong N)
  Set the maximum number of threads.
- is_threadable_abstract ()
  Default constructor. Minimum of one (1) threads.
- is_threadable_abstract (const is_threadable_abstract &a)
  Copy constructor.
- is_threadable_abstract & operator= (const is_threadable_abstract &a)
  Assignment operator.

Protected Attributes

- ulong max_threads

  Maximum number of concurrent threads.

- boost::shared_mutex history_mutex

  Shared mutex for locking during write operations.

- boost::stateful_thread_group threads

  Group of threads.

4.24.1 Constructor & Destructor Documentation

4.24.1.1 is_threadable_abstract::is_threadable_abstract ()[inline]

4.24.1.2 is_threadable_abstract::is_threadable_abstract (const is_threadable_abstract & *a*)[inline]

4.24.2 Member Function Documentation

4.24.2.1 is_threadable_abstract& is_threadable_abstract::operator= (const is_threadable_abstract & a)[inline]

4.24.2.2 ulong *N*)[inline]

4.24.3 Member Data Documentation

4.24.3.1 boost::shared_mutex is_threadable_abstract::history_mutex[mutable, protected]

4.24.3.2 ulong is_threadable_abstract::max_threads[protected]

4.24.3.3 boost::stateful_thread_group is_threadable_abstract::threads[mutable, protected]

The documentation for this class was generated from the following file:

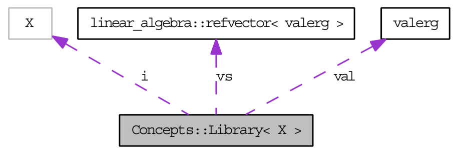
- isthreadableabstract.h

4.25 Concepts::Library< X > Class Template Reference

Concept defines an addressable set of values.

#include <concepts.hh>

Collaboration diagram for Concepts::Library< X >:



Public Member Functions

- BOOST_CONCEPT_USAGE (Library)

Private Attributes

- X i
- valerg val
- ulong index
- ulong mi
- refvector< ulong > rd
- refvector< valerg > vs

template<class X> class Concepts::Library< X >

4.25.1 Member Function Documentation

4.25.1.1 template<class X> Concepts::Library< X >::BOOST_CONCEPT_USAGE (Library< X >)[inline]

Here is the call graph for this function:

4.25.2 Member Data Documentation

4.25.2.1 template<class X> X& Concepts::Library< X >::i[private]

4.25.2.2 template<class X> ulong Concepts::Library< X >::index[private]

4.25.2.3 template<class X> ulong Concepts::Library< X >::mi[mutable, private]

4.25.2.4 template<class X> refvector<ulong> Concepts::Library< X >::rd[mutable, private]

4.25.2.5 template<class X> valerg Concepts::Library< X >::val[private]

4.25.2.6 template<class X> refvector<valerg> Concepts::Library< X >::vs[mutable, private]

The documentation for this class was generated from the following file:

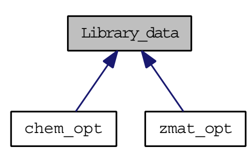
- concepts.hh

4.26 Library_data Class Reference

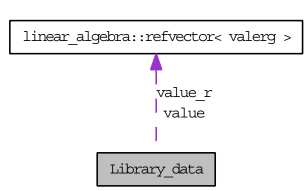
Abstract class for discrete optimizations.

#include <Library_data.hh>

Inheritance diagram for Library_data:



Collaboration diagram for Library_data:



Public Member Functions

- Library_data ()
  Default Constructor.
- Library_data (const Library_data &a)
  Copy constructor.
- virtual ~Library_data ()
  Destructor.
- Library_data & operator= (const Library_data &d)
  Assignment operator.
- virtual ulong get_space_size () const =0
  Compute the size of the optimization space.
- virtual ulong get_bits () const =0
  Compute the number of bits to address the optimization space.
- valerg get_value (ulong i) const
  Retrieve a value for a number.
- void set_Name (const string &A) const
  Sets the name for the purpose of writing files etc.

Public Attributes

- const string & Name_r
  Access to Name (Read-Only).
- const refvector< ulong > & visited_r
  Visited numbers (Read-Only).
- const refvector< valerg > & value_r
  Values of visited numbers (Read-Only).

Protected Member Functions

- virtual valerg compute_property (ulong i) const =0
  Class must have a way to compute values.

Protected Attributes

- refvector< ulong > visited

  Visited numbers.

- refvector< valerg > value

  Values of visited numbers.

- ulong space_size
- bool space_size_computed
- ulong bits
- bool bits_computed
- string Nam

4.26.1 Detailed Description

This class provides the API that guarantees the Concepts::Library concept. The library is considered an access class to a static Library. Therefore, many of its members are mutable since the static library is too large to be stored completely. Instead, entries are computed on the fly and memoized.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 Library_data::Library_data ()[inline]

4.26.2.2 Library_data::Library_data (const Library_data & *a*)[inline]

4.26.2.3 virtual Library_data::∼Library_data ()[inline, virtual]

4.26.3 Member Function Documentation

4.26.3.1 virtual valerg Library_data::compute_property (ulong *i*) const[protected, pure virtual]

Implemented in chem_opt, and zmat_opt.

4.26.3.2 virtual ulong Library_data::get_bits () const[pure virtual]

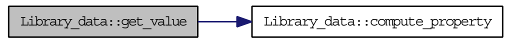
Implemented in chem_opt, and zmat_opt.

4.26.3.3 virtual ulong Library_data::get_space_size () const[pure virtual]

Implemented in chem_opt, and zmat_opt.

4.26.3.4 valerg Library_data::get_value (ulong *i*) const[inline]

Here is the call graph for this function:



4.26.4 Member Data Documentation

4.26.4.1 ulong Library_data::bits[mutable, protected]

4.26.4.2 bool Library_data::bits_computed[mutable, protected]

4.26.4.3 string Library_data::Name[mutable, protected]

4.26.4.4 const string& Library_data::Name_r

4.26.4.5 ulong Library_data::space_size[mutable, protected]

4.26.4.6 bool Library_data::space_size_computed[mutable, protected]

4.26.4.7 refvector<valerg> Library_data::value[mutable, protected]

4.26.4.8 const refvector<valerg>& Library_data::value_r

4.26.4.9 refvector<ulong> Library_data::visited[mutable, protected]

4.26.4.10 const refvector<ulong>& Library_data::visited_r

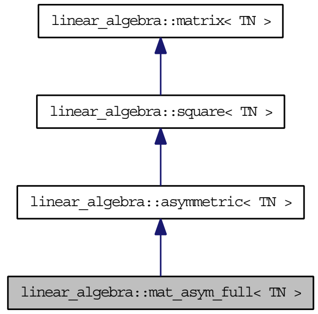
The documentation for this class was generated from the following files:

- Library_data.hh
- Library_data.cc

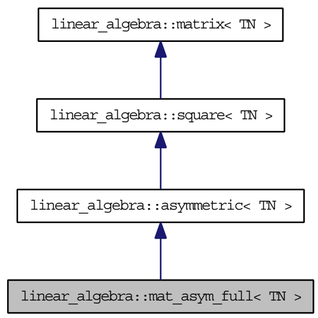
4.27 linear_algebra::mat_asym_full< TN > Class Template Reference

mat_asym_full uses packing for full antisymmetric matrices

Inheritance diagram for linear_algebra::mat_asym_full< TN >:



Collaboration diagram for linear_algebra::mat_asym_full< TN >:



Public Member Functions

- mat_asym_full ()
  Default constructor.
- mat_asym_full (long n)
  Constructor of dimension n with uninitialised values.
- mat_asym_full (long n, const std::vector< TN > &vals)
  Constructor of dimension n with initialisation of values.
- mat_asym_full (long n, refvector< TN > &vals)
  Constructor of dimension n with initialisation of values.
- mat_asym_full (std::istream &IN)
  Constructor to read a stored matrix from a stream. Aimed at retrieving information from a binary save with mat_full::operator>>().
- void operator>> (std::ostream &OUT) const
  Output an asymmetric, full matrix in binary format into a stream. Aimed at file storage.

- const TN & operator() (const long x, const long y) const
  Access operator which checks for integrity in debug mode.

- TN operator() (const long x, const long y)
  Access operator which checks for integrity in debug mode.

- const TN & operator[ ] (const long x) const
  Direct access operator which checks for integrity in debug mode.

- TN & operator[ ] (const long x)
  Direct access operator which checks for integrity in debug mode.

- mat_full< TN > & multiply (const mat_full< TN > &a, mat_full< TN > &r) const
  Multiply by a full matrix.

- mat_full< TN > operator * (const mat_full< TN > &a) const
  Multiply by a full matrix.

- mat_sparse< TN > multiply (const mat_sym_sparse< TN > &b, mat_sparse< TN > &r) const
  Multiply two sparse matrices.

- sparse_vector< TN > & multiply (const sparse_vector< TN > &v, sparse_vector< TN > &r) const
  Multiply an asymmetric full matrix with a sparse vector.

- mat_asym_full< TN > & operator *= (TN a)
  Multiply by a number back into the matrix.

- mat_asym_full< TN > operator * (TN a) const
  Multiply by a number.

- refvector< TN > & multiply (const refvector< TN > &v, refvector< TN > &r) const
  Multiply an asymmetric full matrix with a full vector.

- refvector< TN > operator * (const refvector< TN > &v) const
  Multiply an asymmetric full matrix with a full vector.

- refvector< TN > & multiply (const sparse_vector< TN > &v, refvector< TN > &r) const
  Multiply an asymmetric full matrix with a full vector.

- refvector< TN > operator * (const sparse_vector< TN > &v) const
  Multiply an asymmetric full matrix with a sparse vector.

- mat_asym_full< TN > operator+ (const mat_asym_full< TN > &a) const
  Add an antisymmetric full matrix to an existing one.

- mat_asym_full< TN > & operator+= (const mat_asym_full< TN > &a)
  Add an antisymmetric full matrix to an existing one.

- mat_asym_full< TN > & operator-= (const mat_asym_full< TN > &a)
  Subtract an antisymmetric full matrix to an existing one.

- refvector< TN > extract_col (const long x) const
  Extract a column from an antisymmetric matyrix.

- mat_full< TN > & multiply (const mat_asym_full< TN > &B, mat_full< TN > &r) const
  Mulitply an antisymmetric full matrix with an antisymmetric full matrix.

- mat_full< TN > operator ∗ (const mat_asym_full< TN > &B) const
  Mulitply an antisymmetric full matrix with an antisymmetric full matrix.

- mat_full< TN > operator ∗ (const mat_sym_full< TN > &B) const
  Mulitply an antisymmetric full matrix with a symmetric full matrix.

- mat_full< TN > & multiply (const mat_sym_sparse< TN > &B, mat_full< TN > &r) const
  Mulitply an antisymmetric full matrix with a symmetric full matrix.

- mat_full< TN > operator ∗ (const mat_sym_sparse< TN > &B) const
  Mulitply an antisymmetric full matrix with a symmetric sparse matrix.

- mat_sym_full< TN > aa_$\widehat{()}$ const
  Get the product AA^*.

- void add (const long x, const long y, TN a)

- bool display () const
  Simple display of matrix.

- void set (const long x, const long y, TN value)
  Assignment operation for compatibility with mat_sym_sparse.

- mat_asym_full< TN > & copy (const mat_asym_full< TN > a)
  Copy matrix with depth 1.

- mat_asym_full< TN > & copy (const mat_asym_full< TN > a, const long i)
  Copy matrix with depth i.

- void clear ()

  Reset matrix releasing all associated memory.

- mat_asym_full< TN > & zero ()

  Set matrix to 0, i.e., all elements.

- long size () const

  Returns the size of the vector containing the matrix data.

- mat_asym_full< TN > & utu (const mat_asym_full< TN > &a, mat_asym_full< TN > &R, mat_full< TN > &I) const

  This a symmetric transform SAS.

- mat_asym_full< TN > utu (const mat_asym_full< TN > &a, mat_full< TN > &I) const

  This an antisymmetric transform SAS.

- mat_asym_full< TN > & utu (const mat_asym_full< TN > &a, mat_asym_full< TN > &R) const

  This an antisymmetric transform SAS.

- mat_asym_full< TN > utu (const mat_asym_full< TN > &a) const

  This a symmetric transform SAS.

- mat_asym_full< TN > & utu (const mat_full< TN > &U, mat_asym_full< TN > &R) const

  This is the unitary transform U^*AU.

- mat_asym_full< TN > utu (const mat_full< TN > &U) const

  This is the unitary transform U^tAU.

- mat_asym_full< TN > & uut (const mat_full< TN > &U, mat_asym_full< TN > &R, mat_full< TN > &I) const

  This is the unitary transform UAU^*.

- mat_asym_full< TN > & uut (const mat_full< TN > &U, mat_asym_full< TN > &R) const

  This is the unitary transform UAU^*.

- mat_asym_full< TN > uut (const mat_full< TN > &U) const

  This is the unitary transform UAU^*.

- void gen_mat (const refvector< TN > &a, const refvector< TN > &b)

  Produce the necessary $ij^* - ji^*$.

- mat_full< TN > householder (refvector< TN > &subdiagonal)
  This routine produces the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a full anti-symmetric matrix.
- mat_asym_full< TN > & prune (const TN tol)
  Prune the matrix of negligible values.

Private Attributes

- refvector< TN > _vals
  Vector of values including the diagonal (which by necessity is 0).

4.27.1 Detailed Description

template<class TN> class linear_algebra::mat_asym_full< TN >

The packing is optimised for column access. The matrix is stored as the upper triangle. The lower triangle is the negative of the upper triangle. It is important to note that the diagonal is stored explicitly and may be non-zero. As such, these matrices are not necessarily anti-symmetric, but pseudo antisymmetric in terms of the off diagonal terms.

4.27.2 Constructor & Destructor Documentation

4.27.2.1 template<class TN> linear_algebra::mat_asym_full< TN >::mat_asym_full ()

mat_asym_full uses packing for full antisymmetric matrices Default constructor.

4.27.2.2 template<class TN> linear_algebra::mat_asym_full< TN >::mat_asym_full (long *n*)

4.27.2.3 template<class TN> linear_algebra::mat_asym_full< TN >::mat_asym_full (long *n*, const std::vector< TN > & *vals*)

4.27.2.4 template<class TN> linear_algebra::mat_asym_full< TN >::mat_asym_full (long *n*, refvector< TN > & *vals*)

4.27.2.5 template<class TN> linear_algebra::mat_asym_full< TN >::mat_asym_full (std::istream & *IN*)

4.27.3 Member Function Documentation

4.27.3.1 template<class TN> mat_sym_full< TN > linear_algebra::mat_asym_full< TN >::aa_t () const

Test

4.27.3.2 template<class TN> void linear_algebra::mat_asym_full< TN >::add (const long x, const long y, TN a)

4.27.3.3 template<class TN> void linear_algebra::mat_asym_full< TN >::clear ()

4.27.3.4 template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::copy (const mat_asym_full< TN > a, const long i)

4.27.3.5 template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_full< TN >::copy (const mat_asym_full< TN > a)

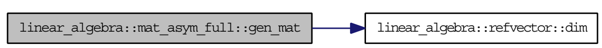
4.27.3.6 template<class TN> bool linear_algebra::mat_asym_full< TN >::display () const

4.27.3.7 template<class TN> refvector< TN > linear_algebra::mat_asym_full< TN >::extract_col (const long x) const

4.27.3.8 template<class TN> void linear_algebra::mat_asym_full< TN >::gen_mat (const refvector< TN > & a, const refvector< TN > & b)

the result is $+= ab^* - ba^*$.

Here is the call graph for this function:



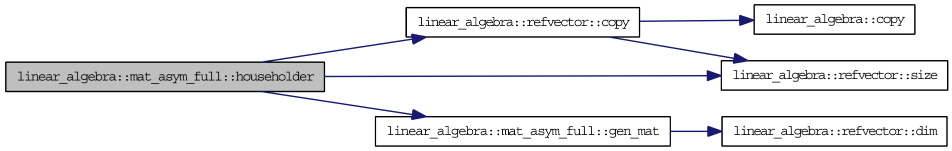
4.27.3.9 template<class TN> mat_full< TN > linear_algebra::mat_asym_full< TN >::householder (refvector< TN > & *subdiagonal*)

The aspired transformation is a multiplication by $(I - vv^*)$ where v is $(\pm e_1\|a_{12}\| + a_{12})/\sqrt{\pm a_{12,1}\|a_{12}\| + \|a_{12}\|^2}$. Notice that we begin counting at 0. The matrix is split up according to $A = \begin{pmatrix} a_{11} & a_{12}^* \\ a_{12} & A_{22} \end{pmatrix}$ We use preconditioning such that the matrix is ordered with lowest off-diagonal value in the low right=hand corner.

$$\begin{pmatrix} a_{11} & \cdots & & \\ & \ddots & & \\ & & \ddots & 0 \\ & & & a_{nn} \end{pmatrix}$$

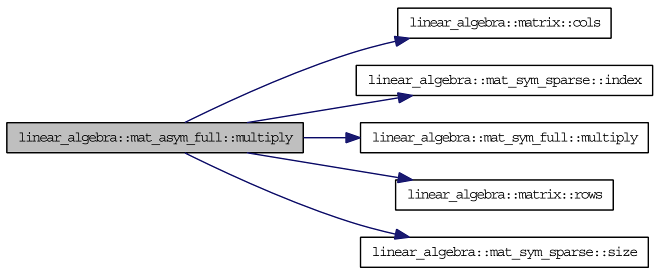
Here is the call graph for this function:

4.27.3.10 template<class TN> mat_full< TN > & linear_algebra::mat_asym_4-full< TN



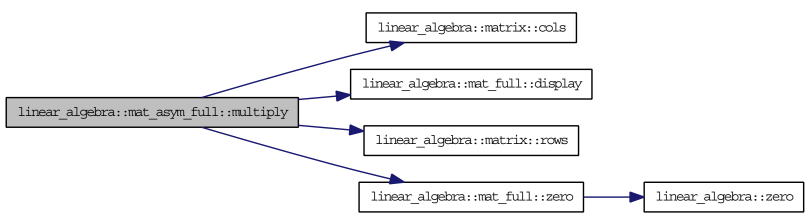
>::multiply (const mat_sym_sparse< TN > & B, mat_full< TN > & r) const

Here is the call graph for this function:



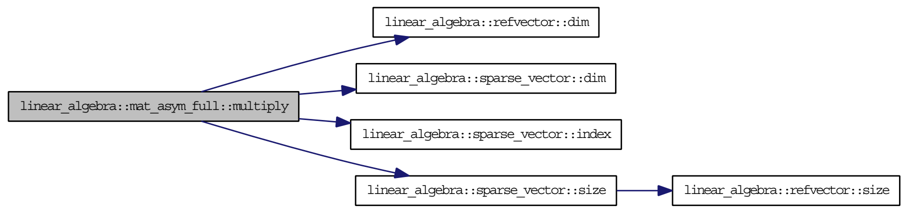
4.27.3.11 template<class TN> mat_full< TN > & linear_algebra::mat_asym_full< TN >::multiply (const mat_asym_full< TN > & B, mat_full< TN > & r) const

Here is the call graph for this function:



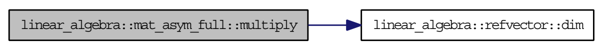
4.27.3.12 template<class TN> refvector< TN > & linear_algebra::mat_asym_full< TN >::multiply (const sparse_vector< TN > & v, refvector< TN > & r) const

Here is the call graph for this function:



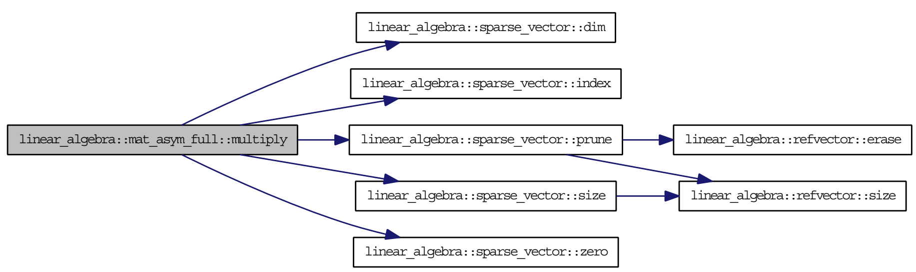
4.27.3.13 template<class TN> refvector< TN > & linear_algebra::mat_asym_full< TN >::multiply (const refvector< TN > & v, refvector< TN > & r) const

Here is the call graph for this function:



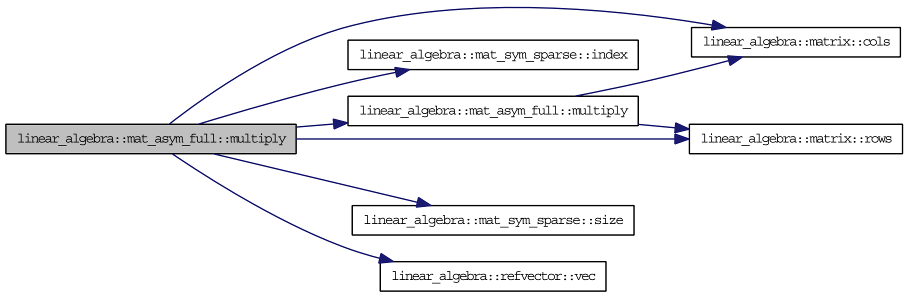
4.27.3.14 template<class TN> sparse_vector< TN > & linear_algebra::mat_asym_full< TN >::multiply (const sparse_vector< TN > & v, sparse_vector< TN > & r) const

Here is the call graph for this function:



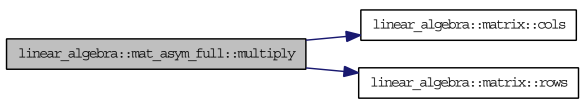
4.27.3.15 template<class TN> mat_sparse< TN > & linear_algebra::mat_asym_full< TN >::multiply (const mat_sym_sparse< TN > & b, mat_sparse< TN > & r) const

Here is the call graph for this function:



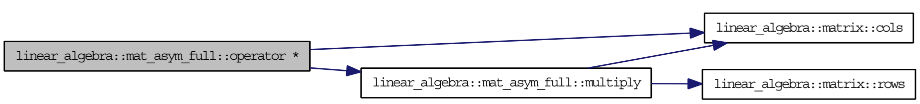
4.27.3.16 template<class TN> mat_full< TN > & linear_algebra::mat_asym_full< TN >::multiply (const mat_full< TN > & *a*, mat_full< TN > & *r*) const

Here is the call graph for this function:



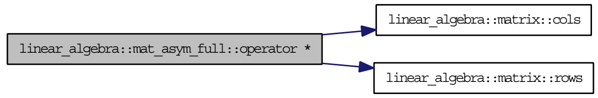
4.27.3.17 template<class TN> mat_full< TN > linear_algebra::mat_asym_full< TN >::operator * (const mat_sym_sparse< TN > & *B*) const

Here is the call graph for this function:



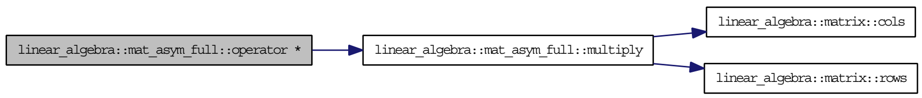
4.27.3.18 template<class TN> mat_full< TN > linear_algebra::mat_asym_full< TN >::operator * (const mat_sym_full< TN > & *B*) const

Here is the call graph for this function:



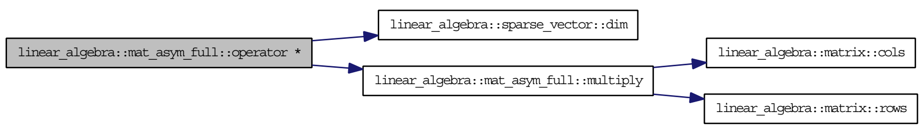
4.27.3.19 template<class TN> mat_full< TN > linear_algebra::mat_asym_full< TN >::operator * (const mat_asym_full< TN > & *B*) const

Here is the call graph for this function:



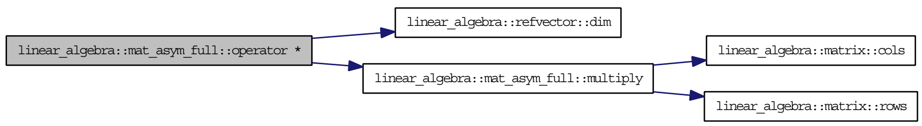
4.27.3.20 template<class TN> refvector< TN > linear_algebra::mat_asym_full< TN >::operator * (const sparse_vector< TN > & *v*) const

Here is the call graph for this function:



4.27.3.21 template<class TN> refvector< TN > linear_algebra::mat_asym_full< TN >::operator * (const refvector< TN > & *v*) const

Here is the call graph for this function:

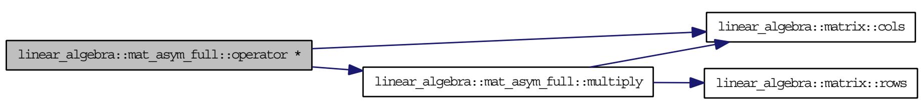


4.27.3.22 template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_full< TN >::operator $*$ (TN a) const

Reimplemented from linear_algebra::asymmetric< TN >.

4.27.3.23 template<class TN> mat_full< TN > linear_algebra::mat_asym_full< TN >::operator $*$ (const mat_full< TN > & a) const

Here is the call graph for this function:



4.27.3.24 template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::operator $*=$ (TN a)

4.27.3.25 template<class TN> TN linear_algebra::mat_asym_full< TN >::operator() (const long x, const long y)

4/27/3/26 template<class TN> const TN & linear_algebra::mat_asym_full< TN >::operator() (const long x, const long y) const[virtual]

Implements linear_algebra::matrix< TN >.

4.27.3.27 template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_full< TN >::operator+ (const mat_asym_full< TN > & a) const

4.37.3.28 template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::operator+= (const mat_asym_full< TN > & a)

4.27.3.29 template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::operator-= (const mat_asym_full< TN > & a)

4.27.3.30 template<class TN> void linear_algebra::mat_asym_full< TN >::operator>> (std::ostream & OUT) const[virtual]

Reimplemented from linear_algebra::matrix< TN >.

4.27.3.31 template<class TN> TN & linear_algebra::mat_asym_full< TN >::operator[] (const long x)

4.27.3.32 template<class TN> const TN & linear_algebra::mat_asym_full< TN >::operator[] (const long x) const

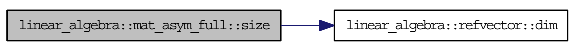
Here is the call graph for this function:

4.27.3.33 template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::prune (const TN *tol*)

4.27.3.34 template<class TN> void linear_algebra::mat_asym_full< TN >::set (const long x, const long y, TN *value*)

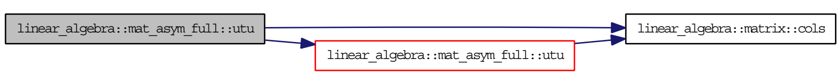
4.27.3.35 template<class TN> long linear_algebra::mat_asym_full< TN >::size () const

Here is the call graph for this function:



4.27.3.36 template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_full< TN >::utu (const mat_full< TN > & U) const

Here is the call graph for this function:

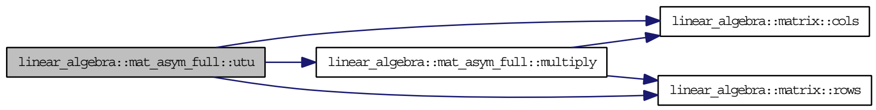


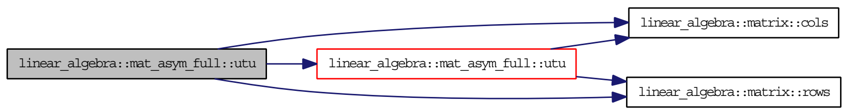
4.27.3.37 template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::utu (const mat_full< TN > & U, mat_asym_full< TN > & R) const

Here is the call graph for this function:

4.27.3.38 template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_full< TN >::utu (const mat_asym_full< TN > & a) const

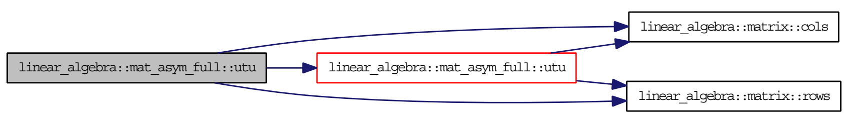
Here is the call graph for this function:





4.27.3.39 template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::utu (const mat_asym_full< TN > & a, mat_asym_full< TN > & R) const

Here is the call graph for this function:



4.27.3.40 template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_full< TN >::utu (const mat_asym_full< TN > & a, mat_full< TN > & I) const

Here is the call graph for this function:

4.27.3.41 template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::utu (const mat_asym_full< TN > & a, mat_asym_full< TN > & R, mat_full< TN > & I) const

Here is the call graph for this function:

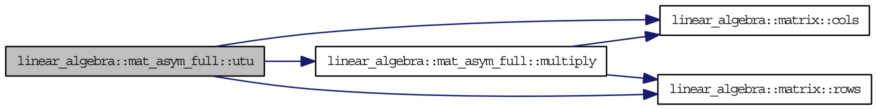
4.27.3.42 template<class TN> mat_asym_full< TN > linear_lgebra::mat_asym_full< TN >::uut (const mat_full< TN > & U) const

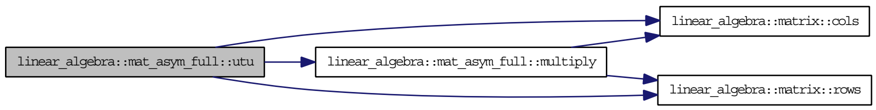
Here is the call graph for this function:

4.27.3.43 template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::uut (const mat_full< TN > & U, mat_asym_full< TN > & R) const

Here is the call graph for this function:

4.27.3.44 template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::uut (const mat_full< TN > & U, mat_asym_full< TN > & R, mat_full< TN > & I) const





Here is the call graph for this function:

4.27.3.45 template<class T> mat_asym_full< T > & linear_algebra::mat_asym_full< T >::zero ()

4.27.4 Member Data Documentation

4.27.4.1 template<class TN> refvector<TN> linear_algebra::mat_asym_full< TN >::_vals[private]

The documentation for this class was generated from the following files:

- include/BCR_CPP_LA/mat_asym_full.decl
- include/BCR_CPP_LA/mat_asym_full.h

4.28 linear_algebra::mat_asym_sparse< TN > Class Template Reference

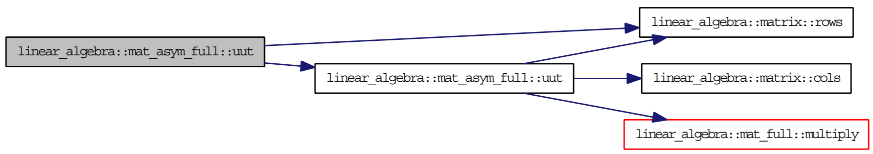
mat_asym_sparse packs sparse column matrices into a comfortable format.

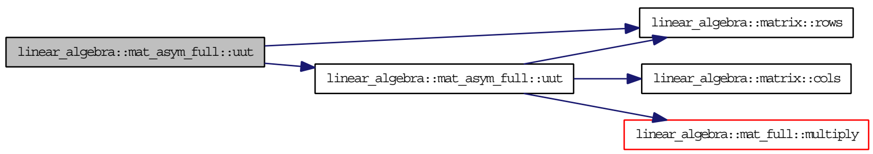
Inheritance diagram for linear_algebra::mat_asym_sparse< TN >:

Collaboration diagram for linear_algebra::mat_asym_sparse< TN >:

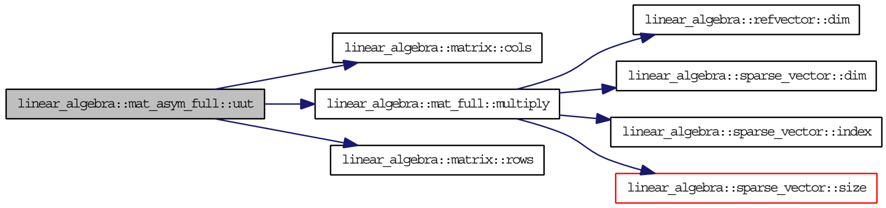
Public Member Functions

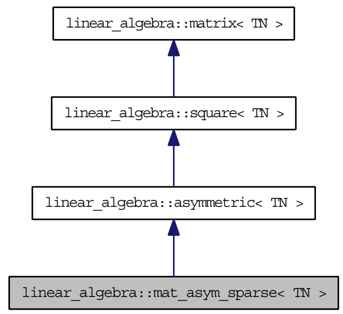
- mat_asym_sparse ()
  Default constructor.
- mat_asym_sparse (long n)
  Constructor of dimension n with uninitialised values.



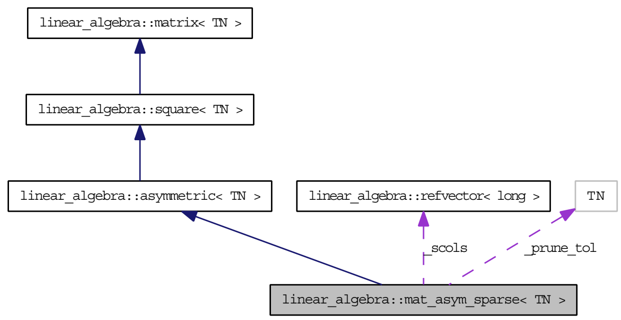


- mat_asym_sparse (long n, const vector< long > &scols, const vector< sparse_vector< TN > > &vals)
  Constructor of dimension n with initialisation of values.
- mat_asym_sparse (long n, const vector< long > &scols, const refvector< sparse_vector< TN > > &vals)
  Constructor of dimension n with initialisation of values.
- mat_asym_sparse (long n, const refvector< long > &scols, const refvector< sparse_vector< TN > > &vals)
  Constructor of dimension n with initialisation of values.
- mat_asym_sparse (long n, const refvector< long > &scols, const vector< sparse_vector< TN > > &vals)
  Constructor of dimension n with initialisation of values.
- mat_asym_spars (istream &IN)
  Constructor to read a stored matrix from a stream. Aimed at retrieving information from a binary save with mat_asym_sparse::operator>>().
- mat_asym_sparse< TN > & operator= (const mat_asym_sparse< TN > &b)
- TN operator() (const long x, const long y) const
  Access operator.
- const sparse_vector< TN > operator[ ] (const long i) const
  Direct access operator.
- sparse_vector< TN > & operator[ ](const long i)
  Direct access operator.
- void set (const long x, const long y, TN value)
  Set operation for safe assignments.





- mat_asym_sparse< TN > operator ∗ (const TN &a) const
  Multiplies an antisymmetric sparse matrix with an object.
- mat_full< TN > operator ∗ (mat_full< TN > A) const
  Multiply by a full matrix.
- mat_full< TN > & multiply (mat_full< TN > A, mat_full< TN > &r) const
  Multiply by a full matrix.
- refvector< TN > & multiply (const refvector< TN > &v, refvector< TN > &r) const
  Multiply a symmetric sparse matrix with a full vector into a full vector./test.
- refvector< TN > operator ∗ (const refvector< TN > &v) const
  Multiply an antisymmetric sparse matrix with a full vector into a full vector./test.
- mat_asym_sparse< TN > & operator ∗= (const TN &x)
  Multiply by an object of TN into left side.
- mat_asym_sparse< TN > & multiply (long x, long y, TN a)
  Multiplies an element by a number.
- mat_full< TN > & multiply (const mat_asym_full< TN > &B, mat_full< TN > &r) const
  Multiply an antisymmetric sparse matrix with a full vector into a full vector./test.
- mat_full< TN > operator ∗ (const mat_asym_full< TN > &B) const
  Multiply a symmetric sparse matrix with a full vector into a full vector./test.



- mat_sym_full< TN > aa_() const
  return a symmetric full matrix=$g \cdot g^*$.
- mat_asym_sparse & operator/=(const TN &x)
  Divide by an object of TN into left side. This is very slow and not advisable.
- sparse_vector< TN > operator * (const sparse_vector< TN > &v) const
  Multiply a sparse symmetric matrix with a sparse vector.
- mat_sparse< TN > operator * (const mat_sparse< TN > &B) const
  Multiply a sparse symmetric matrix with a sparse matrix.
- mat_asym_sparse< TN > & operator+= (const mat_asym_sparse< TN > &B)
  Add a matrix into left object.
- mat_asym_sparse< TN > & operator-= (const mat_asym_sparse< TN > &B)
  Subtract a matrix into left object.
- bool is_col_nonzero (const long x, long &i) const
- bool is_nonzero (long x, long y, long &j, long &k) const
  Checks whether an access is non zero.
- bool add (long x, long y, const TN z)
  Adds a value.
- bool is_empty () const
- long size () const
  Returns size of non-zero entries in _svals.
- void operator>> (ostream &OUT) const
  Output matrix in binary format into a stream. Aimed at file storage.
- mat_asym_sparse< TN > & prune (const TN x=0)
  Deletes all entries that are smaller in magnitude than x.

- TN max_element () const
  Finds a maximum element.
- bool display () const
  Simple display of matrix.
- vector< TN > extract_full_col (const long column, const long xmin, const long xmax) const
  Extracts a block from a column/row.
- vector< TN > extract_full_col (const long column, const long xmin=0) const
  Extracts a block from a column/row.
- mat_asym_full< TN > extract_full_asym_block (long xmin, long xmax) const
  Extracts a full symmetric block.
- mat_asym_full< TN > extract_full_asym_block (long xmin=0) const
  Extracts a full symmetric block.
- mat_sparse< TN > extract_sparse_block (long xmin, long xmax, long ymin, long ymax) const
  Extracts a block from anywhere in the matrix.
- sparse_vector< TN > extract_sparse_col (const long column, const long xmin=0) const
  Extracts a block from a column/row.
- sparse_vector< TN > extract_sparse_lower_col (const long column, const long xmin, const long xmax) const
  Extracts a sparse column up to the diagonal.
- sparse_vector< TN > extract_sparse_lower_col (const long column, const long xmin=0) const
  Extracts a block from a column/row.
- double density () const
  Return the density of non-zero elements.
- void clear ()
  Sets matrix to zero releasing all associated memory.
- mat_asym_sparse< TN > & zero ()
  Sets matrix to zero releasing all associated memory.
- long index (long i) const
  Returns the ith nonzero column number.

- mat_asym_full< TN > & utu (const mat_full< TN > &U, mat_asym_full< TN > &R) const
    This is the unitary transform U^*AU.

- mat_asym_full< TN > utu (const mat_full< TN > &U) const
    This is the unitary transform U^tAU.

Public Attributes

- TN _prune_tol
    Tolerance for multiplication with a sparse vector.

Private Attributes

- refvector< long > _scols
    Vector of column indices.

- refvector< sparse_vector< TN > > _svals
    Vector of sparse columns.

- bool _empty
    State of matrix.

Friends

- class mat_sparse< TN >

- class mat_full< TN >

4.28.1 Detailed Description

template<class TN> class linear_algebra::mat_asym_sparse< TN >

The sparse matrix is packed as a sparse vector of sparse vectors. Only the columns which are not empty are indexed and kept in memory. The sparse vector part takes care of the rows.

4.28.2 Constructor & Destructor Documentation

4.28.2.1 template<class TN> linear_algebra::mat_asym_sparse< TN >::mat_asym_sparse ()

4.28.2.2 template<class TN> linear_algebra::mat_asym_sparse< TN >::mat_asym_sparse (long *n*)

4.28.2.3 template<class TN> linear_algebra::mat_asym_sparse< TN >::mat_asym_sparse (long *n*, const vector< long > & *scols*, const vector< sparse_vector< TN > > & *vals*)

Parameters

> *n* Dimension of matrix.
> *srows* Vector of row indices.
> *scols* Vector of column indices.
> *vals* Vector of values.

Each value has an associated row and column index in srows and scols. Therefore they all have the same size. There is no integrity check.

4.28.2.4 template<class TN> linear_algebra::mat_asym_sparse< TN >::mat_asym_sparse (long *n*, const vector< long > & *scols*, const refvector< sparse_vector< TN > > & *vals*)

> *n* Dimension of matrix.
> *srows* Vector of row indices.
> *scols* Vector of column indices.
> *vals* reference counted Vector of values.

Each value has an associated row and column index in srows and scols. Therefore they all have the same size. There is no integrity check.

4.28.2.5 template<class TN> linear_algebra::mat_asym_sparse< TN >::mat_asym_sparse (long *n*, const refvector< long > & *scols*, const refvector< sparse_vector< TN > > & *vals*)

textbf Parameters:

> *n* Dimension of matrix.
> *srows* Vector of row indices.
> *scols* Reference counted vector of column indices.
> *vals* Reference counted vector of values.

Each value has an associated row and column index in srows and scols. Therefore they all have the same size. There is no integrity check.

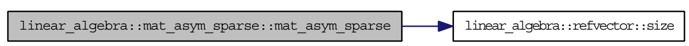
Here is the call graph for this function:

4.28.2.6 template<class TN> linear_algebra::mat_asym_sparse< TN >::mat_asym_sparse (long *n*, const refvector< long > & *scols*, const vector< sparse_vector< TN > > & *vals*)

Parameters:

n Dimension of matrix.
srows Vector of row indices.
scols]Reference counted vector of column indices.
vals Vector of values. Each value has an associated row and column index in srows and scols. Therefore they all have the same size. There is no integrity check.

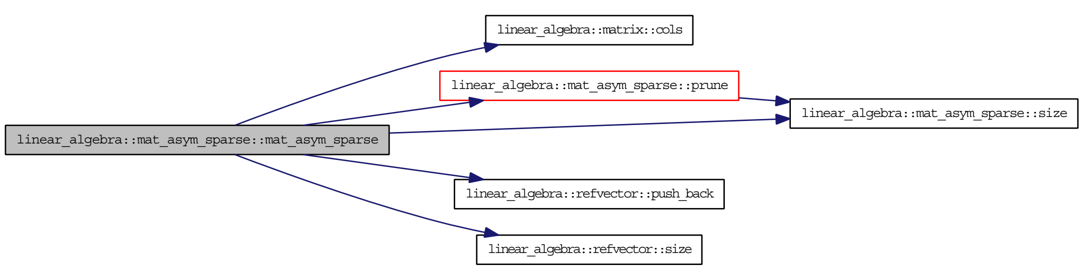
Here is the call graph for this function:



4.28.2.7 template<class TN> linear_algebra::mat_asym_sparse< TN >::mat_asym_sparse (istream & *IN*)

The resultant vector is pruned to default pruning tolerance = 0. This should be completely superfluous but in case a different program not using this class produces the matrix as input we prune.

Here is the call graph for this function:



4.28.3 Member Function Documentation

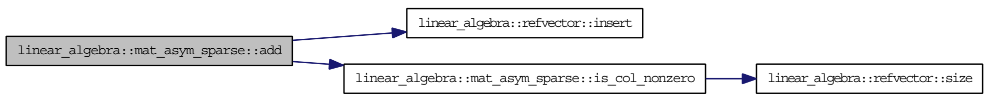
4.28.3.1 template<class TN> mat_sym_full< TN > linear_algebra::mat_asym_sparse< TN >::aa_() const

4.28.3.2 template<class TN> bool linear_algebra::mat_asym_sparse< TN >::add (long x, long y, const TN z)

Parameters:

> z is added to (x,y). If the value is zero the associated vectors are expanded.

Here is the call graph for this function:



4.28.3.3 template<class TN> void linear_algebra::mat_asym_sparse< TN >::clear ()

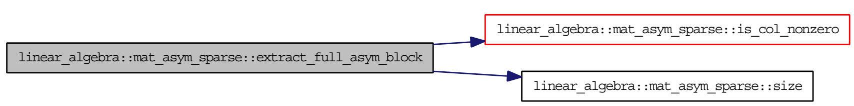
4.28.3.4 template<class TN> double linear_algebra::mat_asym_sparse< TN >::density () const

4.28.3.5 template<class TN> bool linear_algebra::mat_asym_sparse< TN >::display () const

4.28.3.6 template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_sparse< TN >::extract_full_asym_block (long *xmin* = 0) const

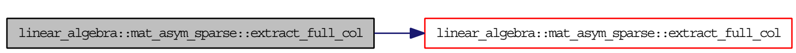
4.28.3.7 template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_sparse< TN >::extract_full_asym_block (long *xmin*, long *xmax*) const

Here is the call graph for this function:



4.28.3.8 template<class TN> vector< TN > linear_algebra::mat_asym_sparse< TN >::extract_full_col (const long *column*, const long *xmin* = 0) const

Here is the call graph for this function:



4.28.3.9 template<class TN> vector< TN > linear_algebra::mat_asym_sparse< TN >::extract_full_col (const long *column*, const long *xmin*, const long *xmax*) const

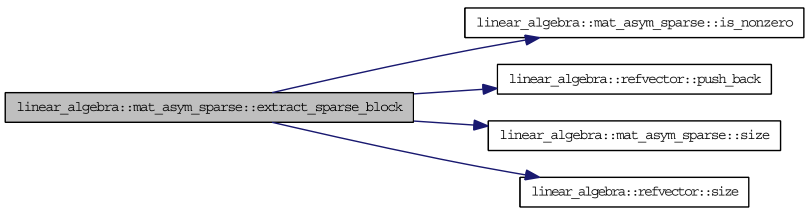
Here is the call graph for this function:

4.28.3.10 template<class TN> mat_sparse< TN > linear_algebra::mat_asym_sparse< TN >::extract_sparse_block (long *xmin*, long *xmax*, long *ymin*, long *ymax*) const

Test

should work, but ...

Here is the call graph for this function:



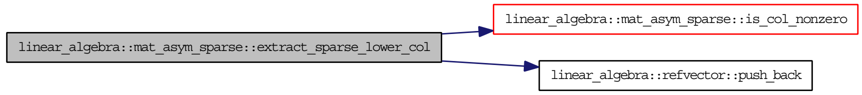
4.28.3.11 template<class TN> sparse_vector< TN > linear_algebra::mat_asym_sparse< TN >::extract_sparse_col (const long *column*, const long *xmin* = 0) const

4.28.3.12 template<class TN> sparse_vector< TN > linear_algebra::mat_asym_sparse< TN >::extract_sparse_lower_col (const long *column*, const long *xmin* = 0) const

Here is the call graph for this function:

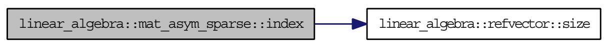
4.28.3.13 template<class TN> sparse_vector< TN > linear_algebra::mat_sym_sparse< TN >::extract_sparse_lower_col (const long *column*, const long *xmin*, const long *xmax*) const

Here is the call graph for this function:



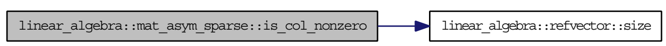
4.28.3.14 template<class TN> long linear_algebra::mat_asym_sparse< TN >::index (long i) const

Here is the call graph for this function:



4.28.3.15 template<class TN> bool linear_algebra::mat_asym_sparse< TN >::is_col_nonzero (const long x, long & i) const

Here is the call graph for this function:



4.28.3.16 template<class TN> bool linear_algebra::mat_asym_sparse< TN >::is_empty () const

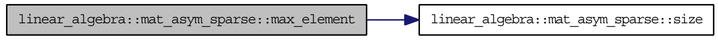
4.28.3.17 template<class TN> bool linear_algebra::mat_asym_sparse< TN >::is_nonzero (long x, long y, long & j, long & k) const

Parameters:

j holds the position of (x,y) in _vals. If (x,y) is 0. An insertion would happen here.

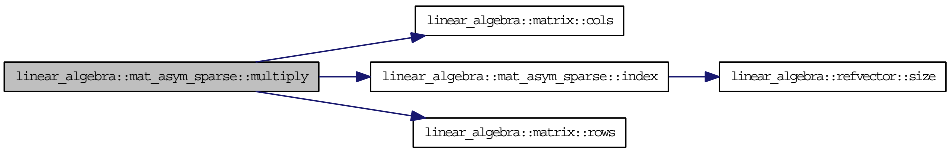
4.28.3.18 template<class TN> TN linear_algebra::mat_asym_sparse< TN >::max_element () const

Here is the call graph for this function:



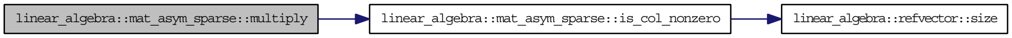
4.28.3.19 template<class TN> mat_full< TN > & linear_algebra::mat_asym_sparse< TN >::multiply (const mat_asym_full< TN > & *B*, mat_full< TN > & *r*) const

Here is the call graph for this function:



4.28.3.20 template<class TN> mat_asym_sparse< TN > & linear_algebra::mat_asym_sparse< TN >::multiply (long *x*, long *y*, TN *a*)

Here is the call graph for this function:



4.28.3.21 template<class TN> refvector< TN > & linear_algebra::mat_asym_sparse< TN >::multiply (const refvector< TN > & *v*, refvector< TN > & *r*) const

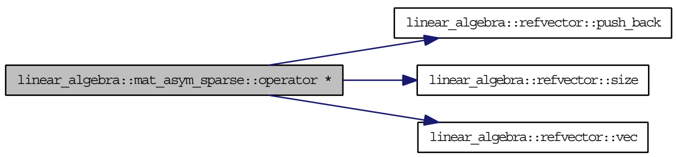
Here is the call graph for this function:

4.28.3.22 template<class TN> mat_full< TN > & linear_algebra::mat_asym_sparse< TN >::multiply (mat_full< TN > A, mat_full< TN > & r) const

Test

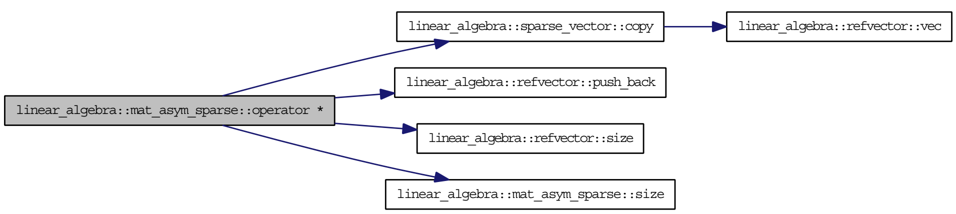
4.28.3.23 template<class TN> mat_sparse< TN > linear_algebra::mat_asym_sparse< TN >::operator $*$ (const mat_sparse< TN > & B) const

Here is the call graph for this function:



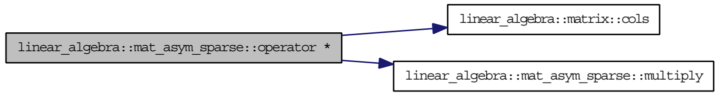
4.28.3.24 template<class TN> sparse_vector< TN > linear_algebra::mat_asym_sparse< TN >::operator $*$ (const sparse_vector< TN > & v) const

Here is the call graph for this function:



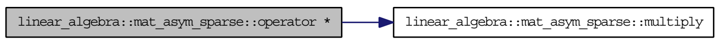
4.28.3.25 template<class TN> mat_full< TN > linear_algebra::mat_asym_sparse< TN >::operator $*$ (const mat_asym_full< TN > & B) const

Here is the call graph for this function:



4.28.3.26 template<class TN> refvector< TN > linear_algebra::mat_asym_sparse< TN >::operator ∗ (const refvector< TN > & v) const

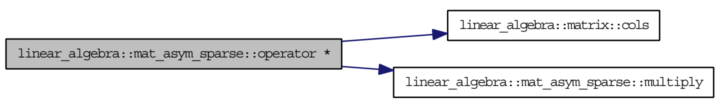
Here is the call graph for this function:



4.28.3.27 template<class TN> mat_full< TN > linear_algebra::mat_asym_sparse< TN >::operator ∗ (mat_full< TN > A) const

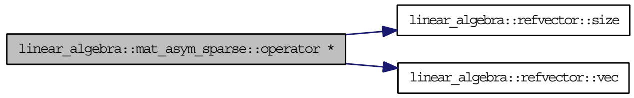
Test

Here is the call graph for this function:



4.28.3.28 template<class TN> mat_asym_sparse< TN > linear_algebra::mat_asym_sparse< TN >::operator ∗ (const TN & a) const

Here is the call graph for this function:

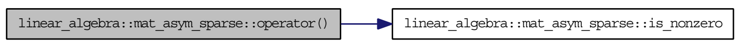


4.28.3.29 template<class TN> mat_asym_sparse< TN > & linear_algebra::mat_asym_sparse< TN >::operator *= (const TN & x)

4.28.3.30 template<class TN> TN linear_algebra::mat_asym_sparse< TN >::operator() (const long x, const long y) const[virtual]

Implements linear_algebra::matrix< TN >.

Here is the call graph for this function:



4.28.3.31 template<class TN> mat_asym_sparse<TN>& linear_algebra::mat_asym_sparse< TN >::operator+= (const mat_asym_sparse< TN > & B)

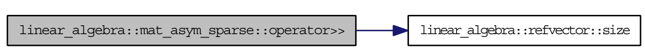
4.28.3.32 template<class TN> mat_asym_sparse<TN>& linear_algebra::mat_asym_sparse< TN >::operator-= (const mat_asym_sparse< TN > & B)

4.28.3.33 template<class TN> mat_asym_sparse< TN > & linear_algebra::mat_asym_sparse< TN >::operator/= (const TN & x)

4.28.3.34 template<class TN> mat_asym_sparse< TN > & linear_algebra::mat_asym_sparse< TN >::operator= (const mat_asym_sparse< TN > & b)

4.28.3.35 template<class TN> void linear_algebra::mat_asym_sparse< TN >::operator>> (ostream & OUT) const

Here is the call graph for this function:

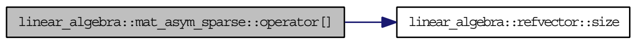


4.28.3.36 template<class TN> sparse_vector< TN > & linear_algebra::mat_asym_sparse< TN >::operator[] (const long i)

Here is the call graph for this function:

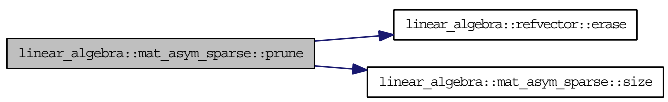
4.28.3.37 template<class TN> const sparse_vector< TN > linear_algebra::mat_asym_sparse< TN >::operator[] (const long i) const

Here is the call graph for this function:



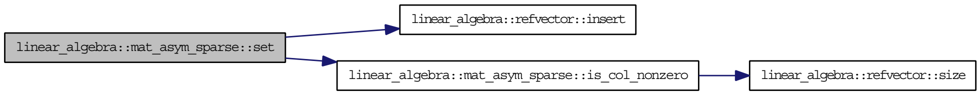
4.28.3.38 template<class TN> mat_asym_sparse< TN > & linear_algebra::mat_asym_sparse< TN >::prune (const TN x = 0)

Here is the call graph for this function:



4.28.3.39 template<class TN> void linear_algebra::mat_asym_sparse< TN >::set (const long x, const long y, TN *value*)

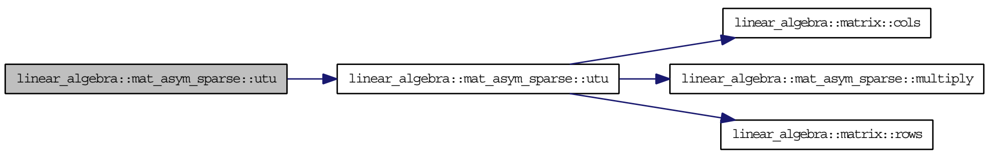
Here is the call graph for this function:



4.28.3.40 template<class TN> long linear_algebra::mat_asym_sparse< TN >::size () const

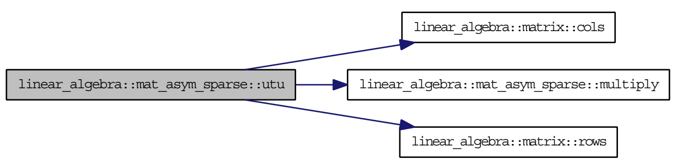
4.28.4.41 template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_sparse< TN >::utu (const mat_full< TN > & U) const

Here is the call graph for this function:



4.28.3.42 template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_sparse< TN >::utu (const mat_full< TN > & U, mat_asym_full< TN > & R) const

Here is the call graph for this function:



4.28.3.43 template<class TN> mat_asym_sparse< TN > & linear_algebra::mat_asym_sparse< TN >::zero ()

4.28.4 Friends and Related Function Documentation

4.28.4.1 template<class TN> friend class mat_full< TN >[friend]

4.28.4.2 template<class TN> friend class mat_sparse< TN >[friend]

4.28.5 Member Data Documentation

4.28.5.1 template<class TN> bool linear_algebra::mat_asym_sparse< TN >::_empty[private]

4.28.5.2 template<class TN> TN linear_algebra::mat_asym_sparse< TN >::_prune_tol

4.28.5.3 template<class TN> refvector<long> linear_algebra::mat_asym_sparse< TN >::_scols[private]

4.28.5.4 template<class TN> refvector<sparse_vector<TN> > linear_algebra::mat_asym_sparse< TN >::_svals[private]

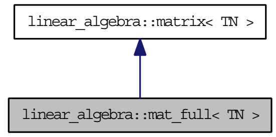
The documentation for this class was generated from the following files:

- include/BCR_CPP_LA/mat_asym_sparse.decl
- include/BCR_CPP_LA/mat_asym_sparse.h

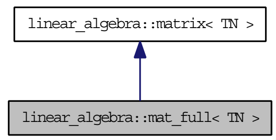
4.29 linear_algebra::mat_full< TN > Class Template Reference

Class of full column matrices.

Inheritance diagram for linear_algebra::mat_full< TN >:



Collaboration diagram for linear_algebra::mat_full< TN >:



Public Member Functions

- mat_full (long n, long m)
  Constructor of n columns and m rows with uninitialised values.
- mat_full (long n, long m, const std::vector< std::vector< TN > > &vals)
  Constructor of n columns and m rows with initialisation of values.
- mat_full (long n, long m, const TN ∗vals[ ])
  Constructor of n columns and m rows with initialisation of values. You had better be sure there is no buffer overrun.
- mat_full (long n, long m, const refvector< std::vector< TN > > &vals)
  Constructor of n columns and m rows with initialisation of values.
- mat_full (long n, long m, const refvector< refvector< TN > > &vals)
  Constructor of n columns and m rows with initialisation of values.
- mat_full (long n, long m, const std::vector< refvector< TN > > &vals)
  Constructor of n columns and m rows with initialisation of values.
- mat_full ()
- mat_full (std::istream &IN)
  Constructor to read a stored matrix from a stream. Aimed at retrieving information from a binary save with mat_full::operator>>().
- ∼mat_full ()

- mat_full< TN > & operator= (mat_full< TN > &a)
  Assignment operator for non const objects.

- mat_full< TN > & operator= (const mat_full< TN > &a)
  Assignment operator for non const objects.

- bool operator== (const mat_full< TN > &b) const
  Compares two full matrices.

- bool same (const mat_full< TN > &b) const

- refvector< TN > & operator[ ] (const long x)
  Return a column through direct access. No integrity checked.

- const refvector< TN > & operator[ ] (const long x) const
  Return a column through direct access. No integrity checked.

- const TN & operator() (const long x, const long y) const
  Return value at (x,y).

- TN & operator() (const long x, const long y)
  Return value at (x,y).

- refvector< TN > & multiply (const sparse_vector< TN > &a, refvector< TN > &r) const
  Multiply a full matrix by a sparse vector.

- refvector< TN > operator ∗ (const sparse_vector< TN > &a) const
  Multiply a full matrix by a sparse vector.

- mat_full< TN > operator ∗ (const TN &a) const
  Multiply a full matrix by a number.

- mat_full< TN > & multiply (const TN &a, mat_full< TN > &r) const
  Multiply a full matrix by a number.

- mat_full< TN > & operator ∗= (const TN &x)
  Multiply by an object of TN into left side.

- mat_full< TN > & operator/= (const TN &x)
  Divide by an object of TN into left side. This is very slow and not advisable.

- mat_full< TN > & operator ∗= (const mat_asym_full< TN > &b)
  Multiply an asymmetric, full matrix from the left into this full matrix.

- mat_full< TN > & operator ∗= (const mat_sym_full< TN > &b)
  Multiply a symmetric, full matrix from the left into this full matrix.

- mat_full< TN > & multiply (const mat_full< TN > &b, mat_full< TN > &r) const
  Multiply two full matrices.
- mat_full< TN > operator ∗ (const mat_full< TN > &b) const
  Multiply two full matrices.
- mat_full< TN > & multiply_t_a (const mat_full< TN > &b, mat_full< TN > &r) const
  Multiply the transpose of a full matrix with a full matrix.
- mat_full< TN > multiply_t_a (const mat_full< TN > &b) const
  Multiply the transpose of a full matrix with a full matrix.
- mat_full< TN > & multiply_a_t (const mat_full< TN > &b, mat_full< TN > &r) const
  Multiply full matrix with the transpose of a full matrix.
- mat_full< TN > multiply_a_t (const mat_full< TN > &b) const
  Multiply full matrix with the transpose of a full matrix.
- mat_full< TN > & multiply (const mat_sym_full< TN > &b, mat_full< TN > &r) const
  Multiply a full matrix by a symmetric full matrix.
- mat_full< TN > operator ∗ (const mat_sym_full< TN > &b) const
  Multiply a full matrix by a symmetric full matrix.
- mat_full< TN > & multiply (const mat_asym_full< TN > &b, mat_full< TN > &r) const
  Multiply a full matrix by an antisymmetric full matrix.
- mat_full< TN > operator ∗ (const mat_asym_full< TN > &b) const
  Multiply a full matrix by an antisymmetric full matrix.
- mat_full< TN > & multiply (const mat_sym_sparse< TN > &b, mat_full< TN > &r) const
  Multiply a full matrix by a symmetric sparse matrix.
- mat_full< TN > operator ∗ (const mat_sym_sparse< TN > &b) const
  Multiply a full matrix by a symmetric sparse matrix.
- template<class TN2> refvector< TN > & multiply (const refvector< TN2 > &a, refvector< TN > &r) const
  Multiply a full matrix by a full vector.

- template<class TN2> refvector< TN > operator * (const refvector< TN2 > &a) const

  Multiply a full matrix by a full vector.

- mat_full< TN > & operator *= (const mat_full< TN > &b)

  *Multiplies the **left** hand side from the { left } by the **right** hand side.*

- mat_full< TN > & operator+= (const mat_full< TN > &b)

  Add two matrices.

- mat_full< TN > & operator+= (const mat_sym_full< TN > &b)

  Add two matrices.

- mat_full< TN > & operator-= (const mat_full< TN > &b)

  Subtract two matrices.

- mat_full< TN > & operator-= (const mat_sym_full< TN > &b)

  Add two matrices.

- mat_full< TN > operator+ (const mat_full< TN > &b) const

  Add two matrices.

- mat_full< TN > operator- (const mat_full< TN > &b) const

  Subtract two matrices.

- mat_full< TN > & copy (const mat_full< TN > &a)

  Copy matrix with full depth.

- mat_full< TN > & copy (const mat_full< TN > &a, const long i)

  Copy matrix with depth i.

- mat_full< TN > & copy (mat_full< TN > &a, const long i)

  Copy matrix with depth i.

- matrix< TN > & copy (const matrix< TN > &a)

  Copy matrix with full depth.

- matrix< TN > & copy (const matrix< TN > &a, long i)

  Copy matrix with depth i.

- matrix< TN > & copy (matrix< TN > &a, long i)

  Copy matrix with depth i.

- bool add(const long x, const long y, const TN z)

  Adds a value.

- void set (const long x, const long y, TN z)
  Set a value.

- bool display () const
  Simple display of matrix.

- bool display (std::ostream &os) const
  Simple display of matrix.

- mat_sym_full< TN > sym_upper_triangle () const
  Returns a mat_sym_full object of the upper triangle of a full matrix.

- mat_sym_full< TN > & sym_upper_triangle (mat_sym_full< TN > &r) const
  Returns a mat_sym_full object of the upper triangle of a full matrix.

- mat_asym_full< TN > asym_upper_triangle () const
  Returns a mat_asym_full object of the upper triangle of a full matrix.

- mat_sym_full< TN > sym_lower_triangle () const
  Returns a mat_sym_full object of the lower triangle of a full matrix.

- mat_sym_full< TN > & sym_lower_triangle (mat_sym_full< TN > &r) const
  Returns a mat_sym_full object of the lower triangle of a full matrix.

- mat_full< double > & gauss_elim (mat_full< TN > &T)
  Do half a householder and then do gaussian elimination for a left inverse.

- mat_full< TN > transpose () const
  Transposes the full matrix.

- mat_full< TN > & transpose (mat_full< TN > &t) const
  Transposes the full matrix.

- mat_full< TN > & grow (long n, long m)
  Extend the matrix by n columns and m rows.

- void clear ()
  Release the associated memory.

- mat_full< TN > & prune (TN a)
  Prune the matrix of negligible values.

- mat_full< TN > & zero ()
  Set the matrix to zero.

- mat_full< TN > & one ()
  Set the matrix to the identity.

Protected Attributes

- refvector< refvector< TN > > _scols
  < *Vector of columns.*

4.29.1 Detailed Description

template<class TN> class linear_algebra::mat_full< TN >

'column' specifies that the values are ordered for speedy retrieval per column.

4.29.2 Constructor & Destructor Documentation

4.29.2.1 template<class TN> linear_algebra::mat_full< TN >::mat_full (long *n*, long *m*)

'column' specifies that the values are ordered for speedy retrieval per column.

4.29.2.2 template<class TN> linear_algebra::mat_full< TN >::mat_full (long *n*, long *m*, const std::vector< std::vector< TN > > & *v*als)

4.29.2.3 template<class TN> linear_algebra::mat_full< TN >::mat_full (long *n*, long *m*, const TN ∗ *vals*[ ])

Parameters:

n Columns of matrix.
m Rows of matrix.
vals array of columns.

Each column is a full array.

Each value has an associated row and column index in srows and scols. Therefore they all have the same size.

4.29.2.4 template<class TN> linear_algebra::mat_full< TN >::mat_full (long *n*, long *m*, const refvector< std::vector< TN > > & *vals*)

4.29.2.5 template<class TN> linear_algebra::mat_full< TN >::mat_full (long *n*, long *m*, const refvector< refvector< TN > > & *vals*)

Parameters:

n Dimension of matrix.

scols Reference counted vector of column indices.
vals Reference counted vector of columns.

Each column is a full vector.

Each value has an associated row and column index in srows and scols. Therefore they all have the same size.

4.29.2.6 template<class TN> linear_algebra::mat_full< TN >::mat_full (long n, long m, const std::vector< refvector< TN > > & *vals*)

4.29.2.7 template<class TN> linear_algebra::mat_full< TN >::mat_full ()

4.29.2.8 template<class TN> linear_algebra::mat_full< TN >::mat_full (std::istream & *IN*)

4.29.4.9 template<class TN> linear_algebra::mat_full< TN >::∼mat_full ()

4.29.3 Member Function Documentation

4.29.3.1 template<class TN> bool linear_algebra::mat_full< TN >::add (const long x, const long y, const TN z)

Parameters:

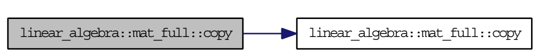
z is added to (x,y). If the value is zero the associated vectors are expanded.

4.29.3.2 template<class TN> mat_asym_full< TN > linear_algebra::mat_full< TN >::asym_upper_triangle () const

4.29.3.3 template<class TN> void linear_algebra::mat_full< TN >::clear ()

4.29.3.4 template<class TN> matrix< TN > & linear_algebra::mat_full< TN >::copy (matrix< TN > & a, long i)

Here is the call graph for this function:



4.29.3.5 template<class TN> matrix< TN > & linear_algebra::mat_full< TN >::copy (const matrix< TN > & a, long i)

Here is the call graph for this function:

4.29.3.6 template<class TN> matrix< TN > & linear_algebra::mat_full< TN >::copy (const matrix< TN > & *a*)

Here is the call graph for this function:

4.29.3.7 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::copy (mat_full< TN > & *a*, const long *i*)

4.29.3.8 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::copy (const mat_full< TN > & *a*, const long *i*)

4.29.3.9 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::copy (const mat_full< TN > & *a*)

4.29.3.10 template<class TN> bool linear_algebra::mat_full< TN >::display (std::ostream & *os*) const

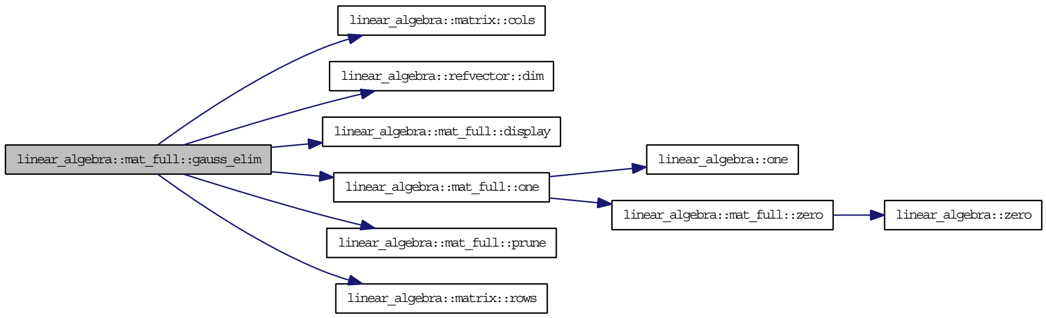
4.29.3.11 template<class TN> bool linear_algebra::mat_full< TN >::display () const

4.29.3.12 template<class TN> mat_full< double > & linear_algebra::mat_full< TN >::gauss_elim (mat_full< TN > & *T*)

Todo

Throw exceptions for ill-conditioned matrices

Here is the call graph for this function:



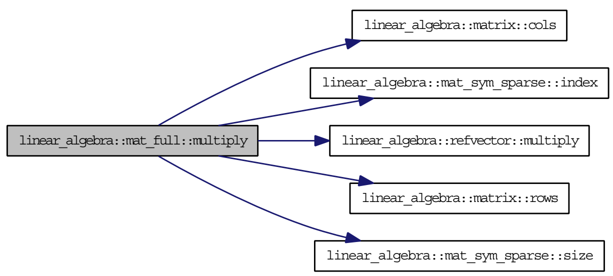
4.29.3.13 template<class T>mat_full< T > & linear_algebra::mat_full< T >::grow (long n, long m)

4.29.3.14 template<class TN> template<class TN2> refvector< TN > & linear_algebra::mat_full< TN >::multiply (const refvector< TN2 > & a, refvector< TN > & r) const

A little care needs to be taken here because any vector for which TN∗TN2 is defined this product is also defined.

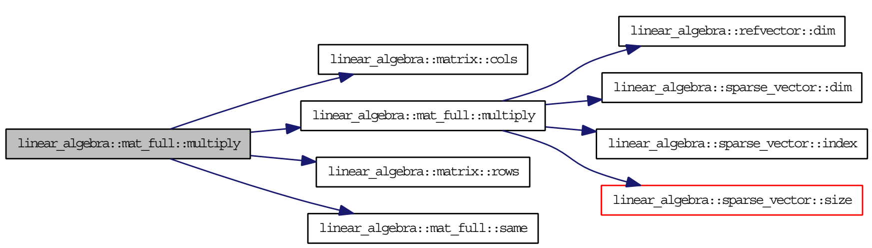
4.29.3.15 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::multiply (const mat_sym_sparse< TN > & b, mat_full< TN > & r) const

Here is the call graph for this function:



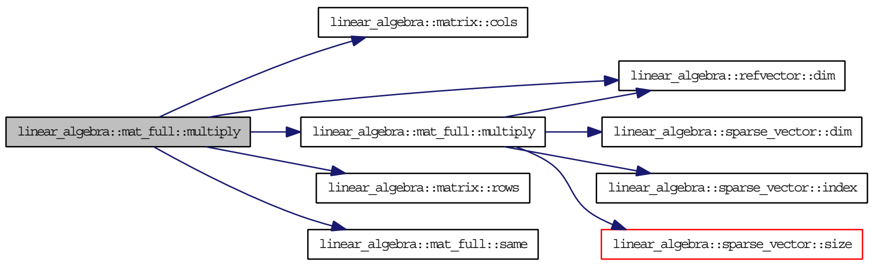
4.29.3.16 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::multiply (const mat_asym_full< TN > & b, mat_full< TN > & r) const

Here is the call graph for this function:



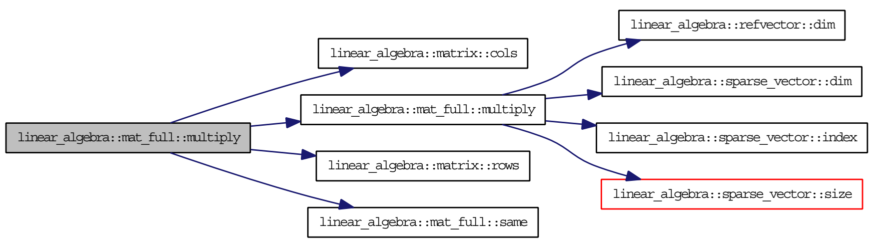
4.29.3.17 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::multiply (const mat_sym_full< TN > &

Here is the call graph for this function:



4.29.3.18 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::multiply (const mat_full< TN > & *b* , mat_full< TN > & *r*) const

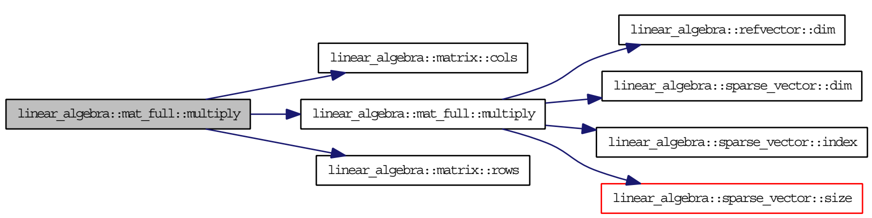
Here is the call graph for this function:



4.29.3.19 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::multiply (const TN & *a*, mat_full< TN > & *r*) const

Test

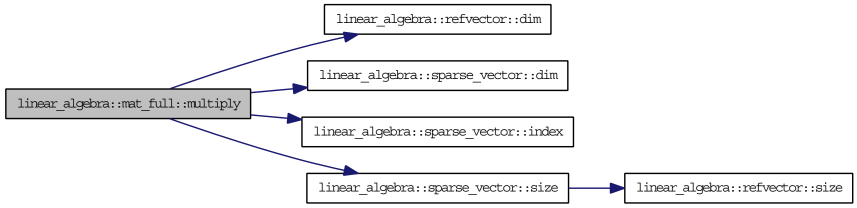
Here is the call graph for this function:



3.29.3.20 template<class TN> refvector< TN > & linear_algebra::mat_full< TN >::multiply (const sparse_vector< TN > & *a*, refvector< TN > & *r*) const

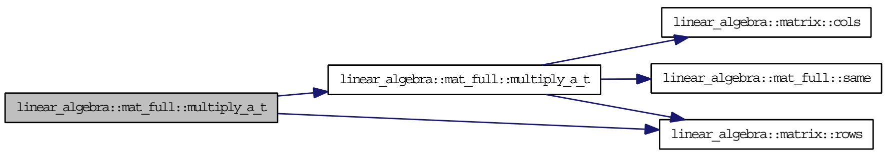
Test

Here is the call graph for this function:



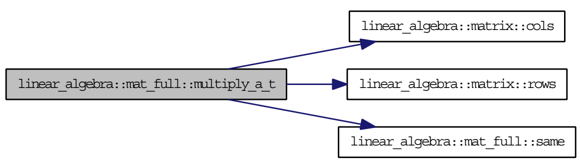
4.29.3.21 template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::multiply_a_t (const mat_full< TN > & *b*) const

Here is the call graph for this function:



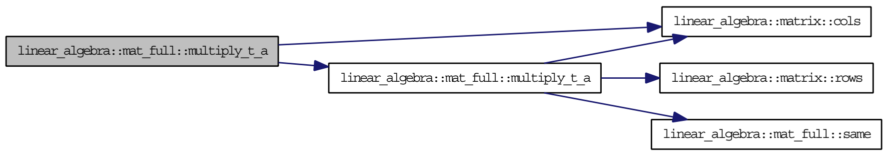
4.29.3.22 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::multiply_a_t (const mat_full< TN > & *b*, mat_full< TN > & *r*) const

Here is the call graph for this function:



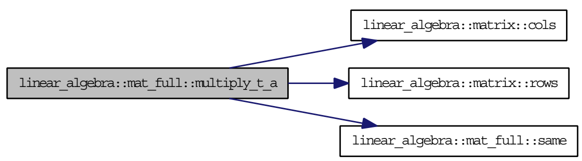
4.29.3.23 template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::multiply_t_a (const mat_full< TN > & *b*) const

Here is the call graph for this function:



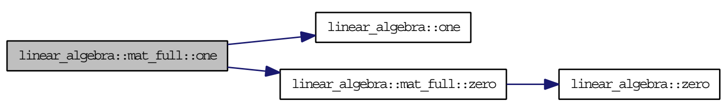
4.29.3.24 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::multiply_t_a (const mat_full< TN > & b, mat_full< TN > & r) const

Here is the call graph for this function:



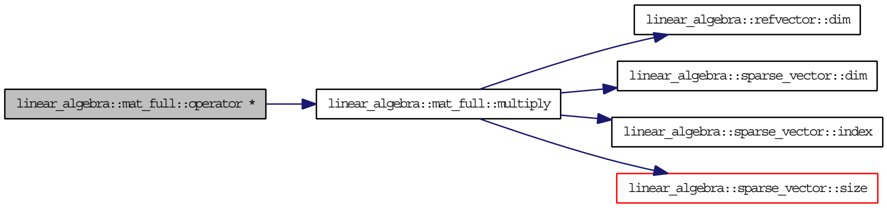
4.29.3.25 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::one ()

Here is the call graph for this function:



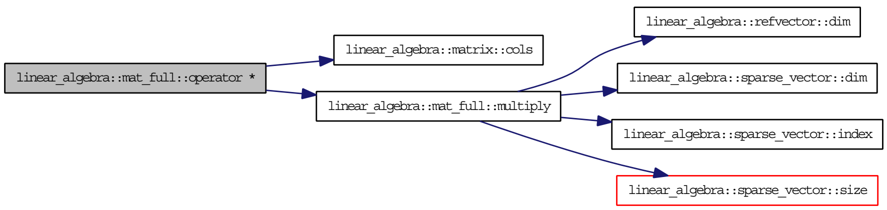
4.29.3.26 template<class TN> template<class TN2> refvector< TN > linear_algebra::mat_full< TN >::operator * (const refvector< TN2 > & a) const

Here is the call graph for this function:



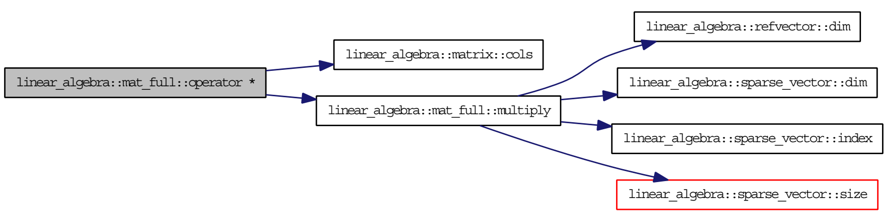
4.29.3.27 template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::operator * (const mat_sym_sparse< TN > & *b*) const

Here is the call graph for this function:



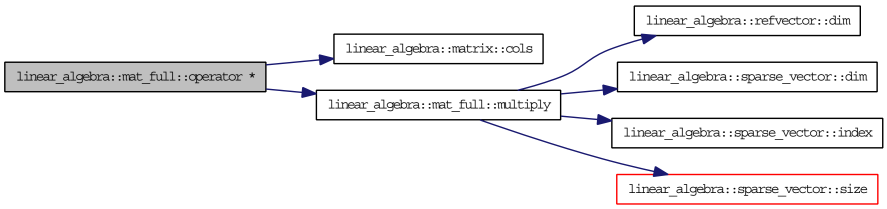
4.29.3.28 template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::operator * (const mat_asym_full< TN > & *b*) const

Here is the call graph for this function:



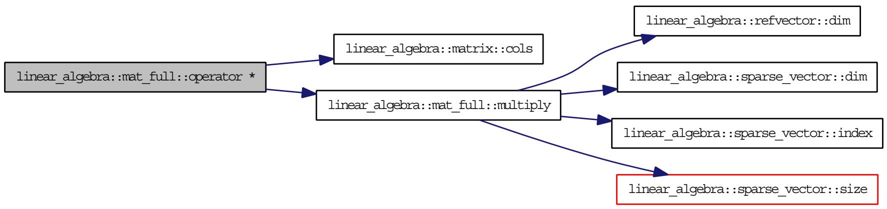
4.29.3.29 template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::operator * (const mat_sym_full< TN > & *b*) const

Here is the call graph for this function:



4.29.3.30 template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::operator * (const mat_full< TN > & *b*) const

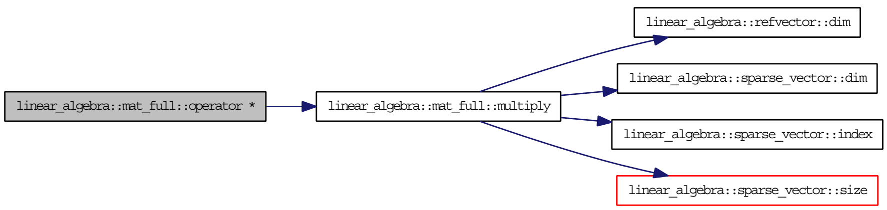
Here is the call graph for this function:



4.29.3.31 template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::operator * (const TN & *a*) const

Test

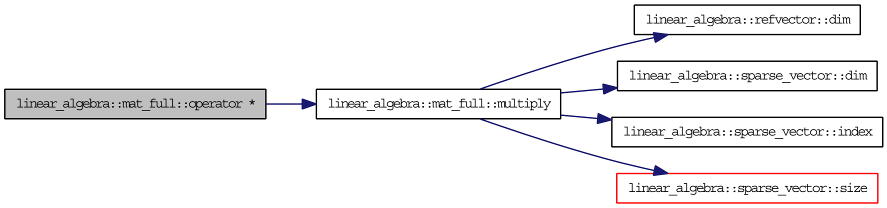
Here is the call graph for this function:



4.29.3.32 template<class TN> refvector< TN > linear_algebra::mat_full< TN >::operator * (const sparse_vector< TN > & *a*) const

Test

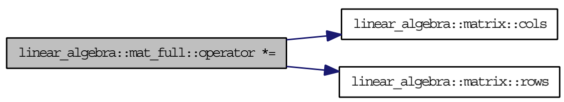
Here is the call graph for this function:



4.29.3.33 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator *= (const mat_full< TN > & *b*)

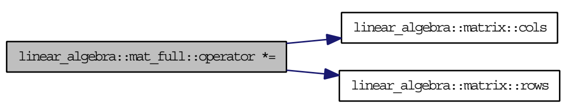
4.29.3.34 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator *= (const mat_sym_full< TN > & *b*)

Here is the call graph for this function:



4.29.3.35 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator *= (const mat_asym_full< TN > & *b*)

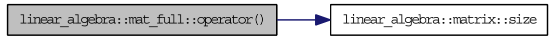
Here is the call graph for this function:



4.29.3.36 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator *= (const TN & *x*)

4.29.3.37 template<class TN> TN & linear_algebra::mat_full< TN >::operator() (const long *x*, const long *y*)

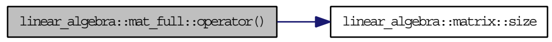
Here is the call graph for this function:



4.29.3.38 template<class TN> const TN & linear_algebra::mat_full< TN >::operator() (const long x, const long *y*) const[virtual]

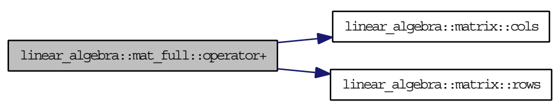
Implements linear_algebra::matrix< TN >.

Here is the call graph for this function:



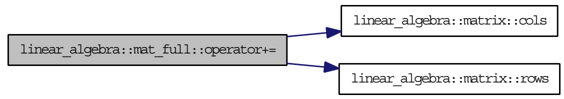
4.29.3.39 template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::operator+ (const mat_full< TN > & *b*) const

Here is the call graph for this function:



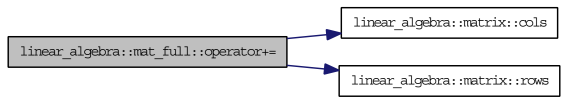
4.29.3.40 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator+= (const mat_sym_full< TN > & *b*)

Here is the call graph for this function:



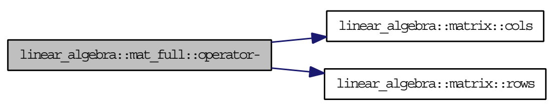
4.29.3.41 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator+= (const mat_full< TN > & *b*)

Here is the call graph for this function:



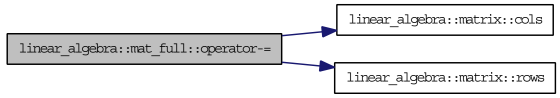
4.29.3.42 template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::operator- (const < TN > & *b*) const

Here is the call graph for this function:



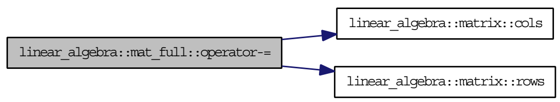
4.29.3.43 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator-= (const mat_sym_full< TN > & *b*)

Here is the call graph for this function:



4.29.3.44 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator-= (const mat_full< TN > & b)

Here is the call graph for this function:



4.29.3.45 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator/= (const TN & x)

4.29.3.46 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator= (const mat_full< TN > & a)

4.29.3.47 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator= (mat_full< TN > & a)

4.29.3.48 template<class TN> bool linear_algebra::mat_full< TN >::operator== (const mat_full< TN > & b) const

4.29.3.49 template<class TN> const refvector< TN > & linear_algebra::mat_full< TN >::operator[] (const long x) const

4.29.3.50 template<class TN> refvector< TN > & linear_algebra::mat_full< TN >::operator[] (const long x)

4.29.3.51 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::prune TN a)

4.29.3.52 template<class TN> bool linear_algebra::mat_full< TN >::same (const mat_full< TN > & b) const

4.29.3.53 template<class TN> void linear_algebra::mat_full< TN >::set (const long x, const long y, TN z)

4.29.3.54 template<class TN> mat_sym_full< TN > & linear_algebra::mat_full< TN >::sym_lower_triangle (mat_sym_full< TN > & r) const

Here is the call graph for this function:

4.29.3.55 template<class TN> mat_sym_full< TN > linear_algebra::mat_full< TN >::sym_lower_triangle () const

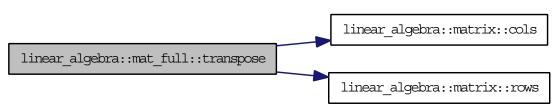
4.29.3.56 template<class TN> mat_sym_full< TN > & linear_algebra::mat_full< TN >::sym_upper_triangle (mat_sym_full< TN > & *r*) const

Here is the call graph for this function:

4.29.3.57 template<class TN> mat_sym_full< TN > linear_algebra::mat_full< TN >::sym_upper_triangle () const

4.29.3.58 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::transpose (mat_full< TN > & *t*) const

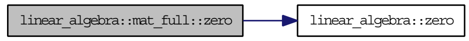
Here is the call graph for this function:



4.29.3.59 template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::transpose () const

4.29.3.60 template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::zero ()

Here is the call graph for this function:



4.29.4 Member Data Documentation

4.29.4.1 template<class TN> refvector<refvector<TN> > linear_algebra::mat_full< TN >::_scols[protected]

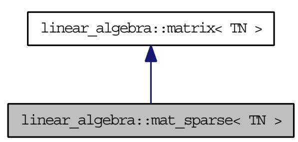
The documentation for this class was generated from the following files:

- mat_full.decl
- mat_full.h
- mat_full.h

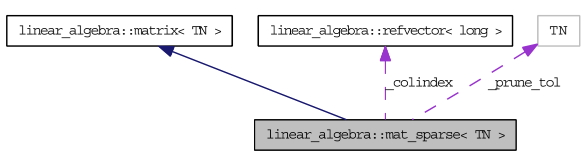
4.30 linear_algebra::mat_sparse< TN > Class Template Reference

Class of sparse column matrices.

Inheritance diagram for linear_algebra::mat_sparse< TN >:



Collaboration diagram for linear_algebra::mat_sparse< TN >:



Public Member Functions

- mat_sparse (long n, long m)
  Constructor of n columns and m rows with uninitialised values.

- mat_sparse (long n, long m, const vector< long > &scols, const vector< sparse_vector< TN > > &vals)

    Constructor of n columns and m rows with initialisation of values.

- mat_sparse (long n, long m, const vector< long > &scols, const refvector< sparse_vector< TN > > &vals)

    Constructor of n columns and m rows with initialisation of values.

- mat_sparse (long n, long m, const refvector< long > &scols, const refvector< sparse_vector< TN > > &vals)

    Constructor of n columns and m rows with initialisation of values.

- mat_sparse (long n, long m, const refvector< long > &scols, const vector< sparse_vector< TN > > &vals)

    Constructor of n columns and m rows with initialisation of values.

- mat_sparse ()

- mat_sparse< TN > & operator= (mat_sparse< TN > &a)

- sparse_vector< TN > & operator[ ] (const long x)

    Return a column through direct access. No integrity checked.

- const sparse_vector< TN > & operator[ ] (const long x) const

    Return a column through direct access. No integrity checked.

- virtual const TN & operator() (const long x, const long y) const

    Return value at (x,y).

- mat_sparse< TN > & operator ∗= (const TN &x)

    Multiply by an object of TN into left side.

- mat_sparse< TN > & operator/= (const TN &x)

    Divide by an object of TN into left side. This is very slow and not advisable.

- mat_sparse< TN > operator ∗ (const mat_sparse< TN > &b) const

    Multiply two sparse matrices.

- sparse_vector< TN > operator ∗ (const sparse_vector< TN > &a) const

    Multiply a sparse matrix by a sparse vector.

- mat_sparse< TN > operator ∗ (const mat_sym_sparse< TN > &b) const

    Multiply a sparse matrix by a sparse symmetric matrix.

- mat_sparse< TN > & operator ∗= (const mat_sparse< TN > &b)

    Multiplies the left hand side from the { left } by the right hand side.

- mat_sparse< TN > & operator *= (const mat_full< TN > &b)
  Multiplies the ***l****eft hand side from the {left } by the* ***r****ight hand side.*
- refvector< TN > & multiply (const refvector< TN > &a, refvector< TN > &b) const
  Multiply a sparse matrix with a full refvector.
- refvector< TN > operator * (const refvector< TN > &a) const
  Multiply a sparse matrix with a full refvector.
- mat_full< TN > & multiply (const mat_full< TN > &a, mat_full< TN > &R) const
  Multiply a sparse matrix with a full matrix.
- mat_full< TN > operator * (const mat_full< TN > &a) const
  Multiply a sparse matrix with a full matrix.
- mat_sparse< TN > & copy (const mat_sparse< TN > &a)
  Copy matrix with deep copy.
- mat_sparse< TN > transpose () const
  Transpose the sparse matrix.
- void prune (const TN x)
  Prune the matrix.
- void prune ()
- bool is_nonzero (long x, long y, long &j, long &k) const
  Checks whether an access is non zero.
- bool is_col_nonzero (const long x, long &i) const
  Checks whether a column is empty.
- bool add (const long x, const long y, const TN z)
  Adds a value.
- bool display () const
  Simple display of matrix.
- vector< TN > extract_full_col (const long column, const long xmin, const long xmax) const
  xtracts a block from a column.
- vector< TN > extract_full_col (const long column, const long xmin) const
  Extracts a block from a column with default max=matrix<TN>_rows.
- sparse_vector< TN > extract_sparse_col (const long column, const long xmin) const
  Extracts a block from a column with default max=matrix<TN>_rows.

- sparse_vector< TN > extract_sparse_col (const long column, const long xmin, const long xmax) const
  Extracts a sparse block from a column.

- mat_sym_sparse< TN > sym_upper_triangle () const
  Returns a mat_sym_sparse object of the upper triangle of a sparse matrix.

- void erase (long c)
  Deletes described column from the matrix. Direct access deletion.

- double density () const
  Return the density of non-zero elements.

- long size () const
  Returns the number of nonzero columns.

- long index (const long x) const
  Returns the column of a given index.

- void set (const long x, const long y, TN z)
  Set a value.

- void clear ()
  Sets the matrix to zero releasing all associated memory.

- mat_sparse< TN > & zero ()
  Sets the matrix to zero releasing all associated memory.

Public Attributes

- TN _prune_tol
  Pruning tolerance.

Protected Attributes

- refvector< sparse_vector< TN > > _scols
  Vector of columns.

- refvector< long > _colindex
  Index of nonzero columns.

- bool _empty
  State of matrix.

Friends

- class mat_sym_sparse< TN >

4.30.1 Detailed Description

template<class TN> class linear_algebra::mat_sparse< TN >

'column' specifies that the values are ordered for speedy retrieval per column.

4.30.2 Constructor & Destructor Documentation

4.30.2.1 template<class TN> linear_algebra::mat_sparse< TN >::mat_sparse (long *n*, long *m*)

4.30.2.2 template<class TN> linear_algebra::mat_sparse< TN >::mat_sparse (long *n*, long *m*, const vector< long > & *scols*, const vector< sparse_vector< TN > > & *vals*)

n Dimension of matrix.
scols Vector of column indices.
vals Vector of columns. Each column is a sparse vector.

Each value has an associated row and column index in srows and scols. Therefore they all have the same size.

4.30.2.3 template<class TN> linear_algebra::mat_sparse< TN >::mat_sparse (long *n*, long *m*, const vector< long > & *scols*, const refvector< sparse_vector< TN > > & *vals*)

Parameters:

n Dimension of matrix.
scols Vector of column indices.
vals Reference counted vector of columns. Each column is a sparse vector.

Each value has an associated row and column index in srows and scols. Therefore they all have the same size.

4.30.2.4 template<class TN> linear_algebra::mat_sparse< TN >::mat_sparse (long *n*, long *m*, const refvector< long > & *scols*, const refvector< sparse_vector< TN > > & *vals*)

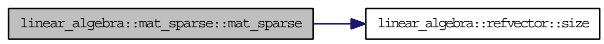
Parameters:

n Dimension of matrix.

scols Reference counted vector of column indices.
vals Reference counted vector of columns. Each column is a sparse vector.

Each value has an associated row and column index in srows and scols. Therefore they all have the same size.

Here is the call graph for this function:



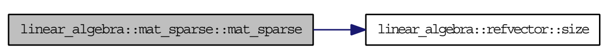
4.30.2.5 template<class TN> linear_algebra::mat_sparse< TN >::mat_sparse (long n, long m, const refvector< long > & *scols*, const vector< sparse_vector< TN > > & *vals*)

Parameters:

n Dimension of matrix.
scols Reference counted Vector of column indices.
vals Vector of columns. Each column is a sparse vector.

Each value has an associated row and column index in srows and scols. Therefore they all have the same size.

Here is the call graph for this function:



4.30.2.6 template<class TN> linear_algebra::mat_sparse< TN >::mat_sparse ()

4.30.3 Member Function Documentation

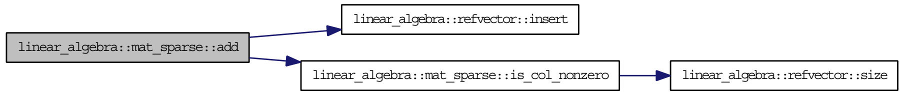
4.30.3.1 template<class TN> bool linear_algebra::mat_sparse< TN >::add (const long x, const long y, const TN z)

Parameters:

z is added to (x,y). If the value is zero the associated vectors are expanded.

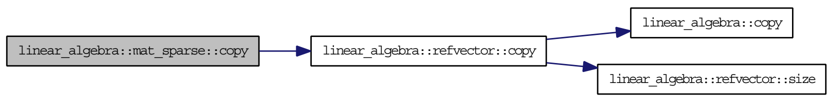
Here is the call graph for this function:

4.30.3.2 template<class TN> void linear_algebra::mat_sparse< TN >::clear ()



4.30.3.3 template<class TN> mat_sparse< TN > & linear_algebra::mat_sparse< TN >::copy (const mat_sparse< TN > & *a*)

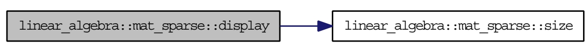
Here is the call graph for this function:



4.30.3.4 template<class TN> double linear_algebra::mat_sparse< TN >::density () const

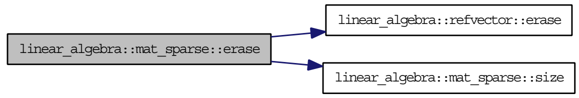
4.30.3.5 template<class TN> bool linear_algebra::mat_sparse< TN >::display () const

Here is the call graph for this function:



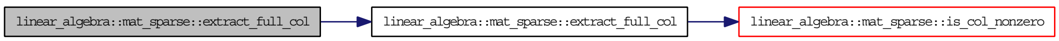
4.30.3.6 template<class TN> void linear_algebra::mat_sparse< TN >::erase (long *c*)

Here is the call graph for this function:



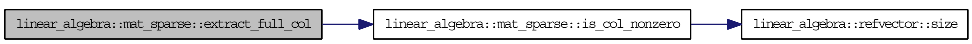
4.30.3.7 template<class TN> vector< TN > linear_algebra::mat_sparse< TN >::extract_full_col (const long *column*, const long *xmin*) const

Here is the call graph for this function:



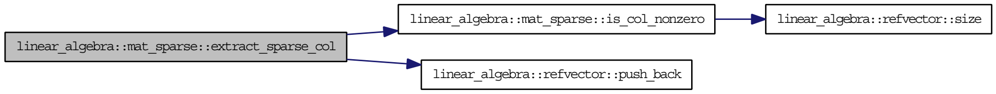
4.30.3.8 template<class TN> vector< TN > linear_algebra::mat_sparse< TN >::extract_full_col (const long *column*, const long *xmin*, const long *xmax*) const

Here is the call graph for this function:



4.30.3.9 template<class TN> sparse_vector< TN > linear_algebra::mat_sparse< TN >::extract_sparse_col (const long *column*, const long *xmin*, const long *xmax*) const

Here is the call graph for this function:



4.30.3.10 template<class TN> sparse_vector< TN > linear_algebra::mat_sparse< TN >::extract_sparse_col (const long *column*, const long *xmin*) const

4.30.3.11 template<class TN> long linear_algebra::mat_sparse< TN >::index (const long *x*) const

4.30.3.12 template<class TN> bool linear_algebra::mat_sparse< TN >::is_col_nonzero (const long *x*, long & *i*) const

Here is the call graph for this function:

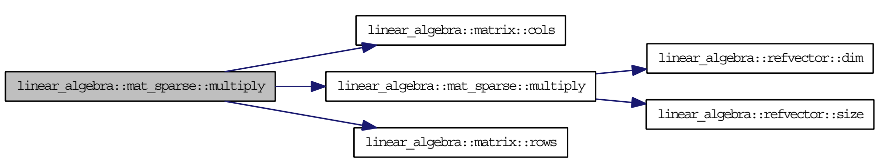
4.30.3.13 template<class TN> bool linear_algebra::mat_sparse< TN >::is_nonzero (long *x*, long *y*, long & *j*, long & *k*) const

Parameters:

j,k hold the position of x,y in _scols,_scols[j]. If (x,y) is 0. An insertion would happen here.

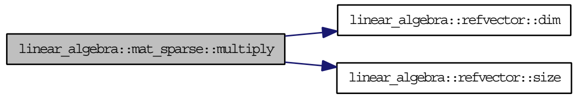
4.30.3.14 template<class TN> mat_full< TN > & linear_algebra::mat_sparse< TN >::multiply (const mat_full< TN > & *a*, mat_full< TN > & *R*) const

Here is the call graph for this function:



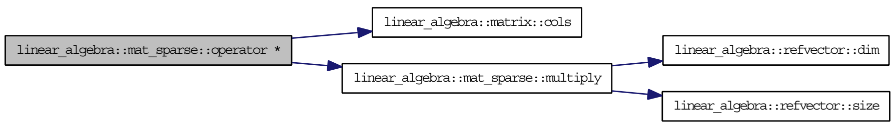
4.30.3.15 template<class TN> refvector< TN > & linear_algebra::mat_sparse< TN >::multiply (const refvector< TN > & *a*, refvector< TN > & *b*) const

Here is the call graph for this function:



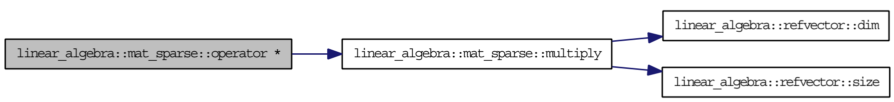
4.30.3.16 template<class TN> mat_full< TN > linear_algebra::mat_sparse< TN >::operator * (const mat_full< TN > & *a*) const

Here is the call graph for this function:



4.30.3.17 template<class TN> refvector< TN > linear_algebra::mat_sparse< TN >::operator * (const refvector< TN > & *a*) const

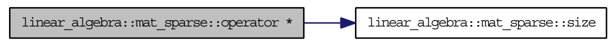
Here is the call graph for this function:



4.30.3.18 template<class TN> mat_sparse< TN > linear_algebra::mat_sparse< TN >::operator * (const mat_sym_sparse< TN > & *b*) const

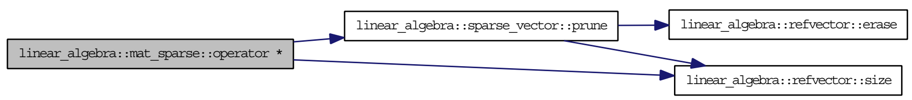
The result is automatically pruned for the default pruning value set in _prune_tol.

Here is the call graph for this function:



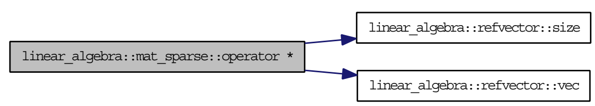
4.30.3.19 template<class TN> sparse_vector< TN > linear_algebra::mat_sparse< TN >::operator * (const sparse_vector< TN > & *a*) const

Here is the call graph for this function:



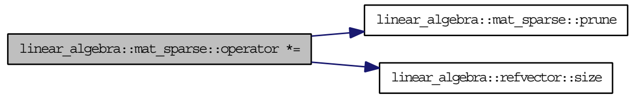
4.30.3.20 template<class TN> mat_sparse< TN > linear_algebra::mat_sparse< TN >::operator * (const mat_sparse< TN > & *b*) const

Here is the call graph for this function:



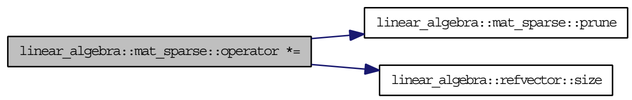
4.30.3.21 template<class TN> mat_sparse< TN > & linear_algebra::mat_sparse< TN >::operator *= (const mat_full< TN > & *b*)

Here is the call graph for this function:



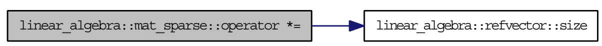
4.30.3.22 template<class TN> mat_sparse< TN > & linear_algebra::mat_sparse< TN >::operator *= (const mat_sparse< TN > & *b*)

Here is the call graph for this function:



4.30.3.23 template<class TN> mat_sparse< TN > & linear_algebra::mat_sparse< TN >::operator *= (const TN & *x*)

Here is the call graph for this function:



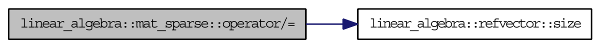
4.30.3.24 template<class TN> const TN & linear_algebra::mat_spars< TN >::operator() (const long *x*, const long *y*) const[virtual]

Implements linear_algebra::matrix< TN >.

Here is the call graph for this function:

4.30.3.25 template<class TN> mat_sparse< TN > & linear_algebra::mat_sparse< TN >::operator/= (const TN & *x*)

Here is the call graph for this function:



4.30.3.26 template<class TN> mat_sparse< TN > & linear_algebra::mat_sparse< TN >::operator= (mat_sparse< TN > & *a*)

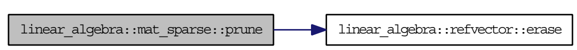
4.30.3.27 template<class TN> const sparse_vector< TN > & linear_algebra::mat_sparse< TN >::operator[ ] (const long *x*) const

4.30.3.28 template<class TN> sparse_vector< TN > & linear_algebra::mat_sparse< TN >::operator[ ] (const long *x*)

4.30.3.29 template<class TN> void linear_algebra::mat_sparse< TN >::prune ()

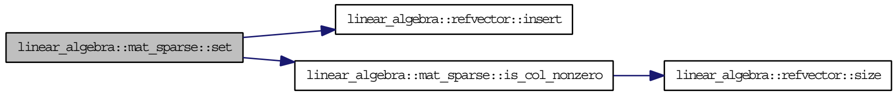
4.30.3.30 template<class TN> void linear_algebra::mat_sparse< TN >::prune (const TN *x*)

Here is the call graph for this function:



4.30.3.31 template<class TN> void linear_algebra::mat_sparse< TN >::set (const long *x*, const long *y*, TN *z*)

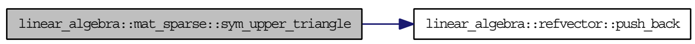
Here is the call graph for this function:



4.30.3.32 template<class TN> long linear_algebra::mat_sparse< TN >::size () const

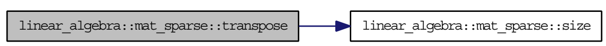
4.30.3.33 template<class TN> mat_sym_sparse< TN > linear_algebra::mat_sparse< TN >::sym_upper_triangle () const

Test
Here is the call graph for this function:



4.30.3.34 template<class TN> mat_sparse< TN > linear_algebra::mat_sparse< TN >::transpose () const

Here is the call graph for this function:



4.30.3.35 template<class TN> mat_sparse< TN > & linear_algebra::mat_sparse< TN >::zero ()

4.30.4 Friends and Related Function Documentation

4.30.4.1 template<class TN> friend class mat_sym_sparse< TN >[friend]

4.30.5 Member Data Documentation

4.30.5.1 template<class TN> refvector<long> linear_algebra::mat_sparse< TN >::_colindex[protected]

4.30.5.2 template<class TN> bool linear_algebra::mat_sparse< TN >::_empty[protected]

4.30.5.3 template<class TN> TN linear_algebra::mat_sparse< TN >::_prune_tol

4.30.5.4 template<class TN> refvector<sparse_vector<TN> > linear_algebra::mat_sparse< TN >::_scols[protected]

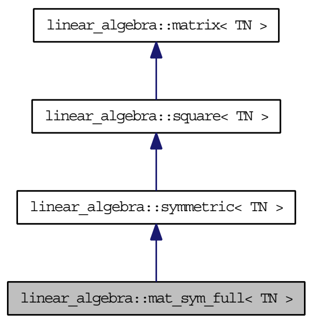
The documentation for this class was generated from the following files:

- include/BCR_CPP_LA/mat_sparse.decl
- include/BCR_CPP_LA/mat_sparse.h

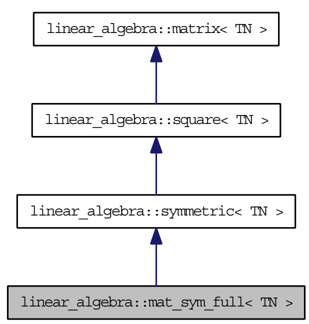
4.31 linear_algebra::mat_sym_full< TN > Class Template Reference

mat_sym_full uses packing for full symmetric matrices

Inheritance diagram for linear_algebra::mat_sym_full< TN >:



Collaboration diagram for linear_algebra::mat_sym_full< TN >:



Public Member Functions

- mat_sym_full ()
  Default constructor.
- mat_sym_full (istream &IN)
  Constructor to read a stored matrix from a stream. Aimed at retrieving information from a binary save with mat_sym_full::operator>>().

- mat_sym_full (long n)

  Constructor of dimension n with uninitialised values.

- mat_sym_full (long n, const vector< TN > &vals)

  Constructor of dimension n with initialisation of values.

- mat_sym_full (long n, refvector< TN > &vals)

  Constructor of dimension n with initialisation of values.

- void operator> > (ostream &OUT) const

  Output matrix in binary format into a stream. Aimed at file storage.

- mat_sym_full< TN > & operator= (mat_sym_full< TN > &a)

  Assignement operator for non-const matrices.

- mat_sym_full< TN > & operator= (const mat_sym_full< TN > &a)

  Assignement operator for non-const matrices.

- const TN & operator() (const long x, const long y) const

  Access operator which checks for integrity in debug mode.

- TN & operator() (const long x, const long y)

  Access operator which checks for integrity in debug mode.

- const TN & operator mbox[] (const long x) const

  Direct access operator which checks for integrity in debug mode.

- TN & operator mbox[] (const long x)

  Direct access operator which checks for integrity in debug mode.

- refvector< TN > operator() (const long column, const long xmin, const long xmax) const

  Extracts a block from a column/row.

- refvector< TN > operator() (const long column) const

  Extracts a column/row from the matrix.

- mat_sym_full< TN > & multiply (long x, long y, TN a)

  Addition operation for compatibility with mat_sym_sparse.

- mat_full< TN > & multiply (const mat_full< TN > &b, mat_full< TN > &r) const

  Multiplies a symmetric full matrix with a full matrix.

- mat_full< TN > operator * (const mat_full< TN > &b) const

- mat_full< TN > & multiply (const mat_sym_full< TN > &b, mat_full< TN > &r) const

  Multiply a symmetric full matrix with a symmetric full matrix.

- mat_full< TN > operator * (const mat_sym_full< TN > &b) const

  Multiply a symmetric full matrix with a symmetric full matrix.

- mat_full< TN > & multiply (const mat_asym_full< TN > &b, mat_full< TN > &r) const

  Multiply a symmetric full matrix with an antisymmetric full matrix.

- mat_full< TN > operator * (const mat_asym_full< TN > &b) const

  Multiply a symmetric full matrix with an antisymmetric full matrix.

- template<class TN2> refvector< TN > & multiply (const sparse_vector< TN2 > &v, refvector< TN > &r) const

  Multiply a symmetric full matrix with a sparse vector.

- template<class TN2> refvector< TN > operator * (const sparse_vector< TN2 > &v) const

  Multiply a symmetric full matrix with a sparse vector.

- template<class TN2> refvector< TN > & multiply (const refvector< TN2 > &v, refvector< TN > &r) const

  Multiply a symmetric full matrix with a full vector.

- template<class TN2> refvector< TN > operator * (const refvector< TN2 > &v) const

  Multiply a symmetric full matrix with a full vector.

- template<class TN2> mat_sym_full< TN > operator *= (const TN2 &a)

  Multiply a symmetric full matrix with an element.

- template<class TN2> mat_sym_full< TN > & operator/= (const TN2 &a)

  Divide a symmetric full matrix by an element.

- mat_sym_full< TN > & multiply (const TN &a, mat_sym_full< TN > &r) const

  Multiply a symmetric full matrix with an element.

- mat_sym_full< TN > operator * (const TN &a) const

  Multiply a symmetric full matrix with an element.

- mat_sym_full< TN > operator/ (const TN &a) const

  Divide a symmetric full matrix by an element.

- mat_sym_full< TN > & operator+= (const mat_sym_full< TN > &a)
  Add two symmetric matrices.

- mat_sym_full< TN > & operator+= (const mat_sym_sparse< TN > &a)
  Add two symmetric matrices.

- mat_sym_full< TN > & operator-= (const mat_sym_full< TN > &a)
  Subtract two symmetric matrices.

- mat_sym_full< TN > & operator-= (const mat_sym_sparse< TN > &a)
  Subtract two symmetric matrices.

- mat_sym_full< TN > operator+ (const mat_sym_full< TN > &a) const
  Add two symmetric matrices.

- mat_sym_full< TN > operator- (const mat_sym_full< TN > &a) const
  Subtract two symmetric matrices.

- mat_sym_full< TN > operator- (const mat_sym_sparse< TN > &a) const
  Subtract two symmetric matrices, one full one sparse.

- bool display () const
  Simple display of matrix.

- void set (const long x, const long y, TN value)
  Assignment operation for compatibility with mat_sym_sparse.

- bool add (long x, long y, const TN value)
  Addition operation for compatibility with mat_sym_sparse.

- void gen_mat (const refvector< TN > &a, const refvector< TN > &b, TN correction)
  Produce the necessary $ij^* + ji^*$.

- mat_sym_full< TN > unitary_transform (const mat_full< TN > U) const
  Transforms a symmetric full matrix by a unitary matrix. O(n^4) UAU^t.

- mat_full< TN > householder (refvector< TN > &diagonal, refvector< TN > &subdiagonal)
  This routine produces the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.

- mat_sym_full< TN > & copy (const mat_sym_full< TN > &a)
  Copy matrix with full depth.

- mat_sym_full< TN > & copy (const mat_sym_full< TN > &a, const long i)
  Copy matrix with depth i.

- mat_sym_full< TN > & copy mat_sym_full< TN > &a, const long i)
  Copy matrix with depth i.
- symmetric< TN > & copy (const symmetric< TN > &a)
  Copy matrix with depth 1.
- symmetric< TN > & copy (const symmetric< TN > &a, const long i)
  Copy matrix with depth i.
- symmetric< TN > & copy (symmetric< TN > &a, const long i)
  Copy matrix with depth i.
- void clear ()
  Release the memory associated with the matrix.
- mat_sym_full< TN > & uut(const mat_full< TN > &U, mat_sym_full< TN > &R, mat_full< TN > &I) const
  This is the unitary transform UAU^*.
- mat_sym_full< TN > & uut (const mat_full< TN > &U, mat_sym_full< TN > &R) const
  This is the unitary transform UAU^*.
- mat_sym_full< TN > uut (const mat_full< TN > &U) const
  This is the unitary transform UAU^*.
- mat_sym_full< TN > & utu (const mat_sym_full< TN > &a, mat_sym_full< TN > &R, mat_full< TN > &I) const
  This a symmetric transform SAS.
- mat_sym_full< TN > utu (const mat_sym_full< TN > &a, mat_full< TN > &I) const
  This a symmetric transform SAS.
- mat_sym_full< TN > & utu (const mat_sym_full< TN > &a, mat_sym_full< TN > &R) const
  This a symmetric transform SAS.
- mat_sym_full< TN > utu (const mat_sym_full< TN > &a) const
  This a symmetric transform SAS.
- mat_sym_full< TN > & utu (const mat_full< TN > &U, mat_sym_full< TN > &R) const
  This is the unitary transform U^*AU.

- mat_sym_full< TN > & utu (const mat_full< TN > &U, mat_sym_full< TN > &R, refvector< TN > &v) const
  This is the unitary transform U^*AU.

- mat_sym_full< TN > utu (const mat_full< TN > &U) const
  This is the unitary transform $U^\wedge tAU$.

- mat_sym_full< TN > utu (const mat_asym_full< TN > &a, mat_full< TN > &I) const
  This a symmetric transform SAS.

- mat_sym_full< TN > utu (const mat_asym_full< TN > &U) const
  This is the unitary transform $U^\wedge tAU$.

- mat_sym_full< TN > & prune (const TN tol)
  Prune the matrix of negligible values.

- mat_sym_full< TN > & zero ()
  Set the matrix to zero.

- mat_sym_full< TN > & resize (long n)
  Constructor of dimension n with uninitialised values.

Private Attributes

- refvector< TN > _vals
  Vector of values.

4.31.1 Detailed Description

template<class TN> class linear_algebra::mat_sym_full< TN >

The packing is optimised for column access.

4.31.2 Constructor & Destructor Documentation

4.31.2.1 template<class TN> linear_algebra::mat_sym_full< TN >::mat_sym_full ()

4.31.2.2 template<class TN> linear_algebra::mat_sym_full< TN >::mat_sym_full (istream & *IN*)

4.31.2.3 template<class TN> linear_algebra::mat_sym_full< TN >::mat_sym_full (long *n*)

4.31.2.4 template<class TN> linear_algebra::mat_sym_full< TN >::mat_sym_full (long *n*, const vector< TN > & *vals*)

4.31.2.5 template<class TN> linear_algebra::mat_sym_full< TN >::mat_sym_full (long *n*, refvector< TN > & *vals*)

4.31.3 Member Function Documentation

4.31.3.1 template<class TN> bool linear_algebra::mat_sym_full< TN >::add (long *x*, long *y*, const TN *value*)

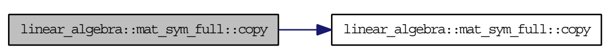
Reimplemented from linear_algebra::symmetric< TN >.

4.31.3.2 template<class TN> void linear_algebra::mat_sym_full< TN >::clear ()

4.31.3.3 template<class TN> symmetric< TN > & linear_algebra::mat_sym_full< TN >::copy (symmetric< TN > & *a*, const long *i*)[virtual]

Implements linear_algebra::symmetric< TN >.

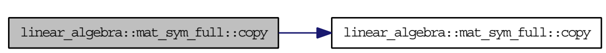
Here is the call graph for this function:



4.31.3.4 template<class TN> symmetric< TN > & linear_algebra::mat_sym_full< TN >::copy (const symmetric< TN > & *a*, const long *i*)[virtual]

Implements linear_algebra::symmetric< TN >.

Here is the call graph for this function:



4.31.3.5 template<class TN> symmetric< TN > & linear_algebra::mat_sym_full< TN >::copy (const symmetric< TN > & *a*)[virtual]

Implements linear_algebra::symmetric< TN >.

Here is the call graph for this function:

4.31.3.6 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::copy (mat_sym_full< TN > & *a*, const long *i*)

4.31.3.7 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::copy (const mat_sym_full< TN > & *a*, const long *i*)

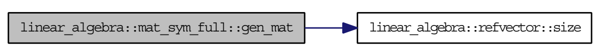
4.31.3.8 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::copy (const mat_sym_full< TN > & *a*)

4.31.3.9 template<class TN> bool linear_algebra::mat_sym_full< TN >::display () const

4.31.3.10 template<class TN> void linear_algebra::mat_sym_full< TN >::gen_mat (const refvector< TN > & *a*, const refvector< TN > & *b*, TN *correction*)

the result is $- = ab^* + ba^*$ (-=ab^ + ba^).

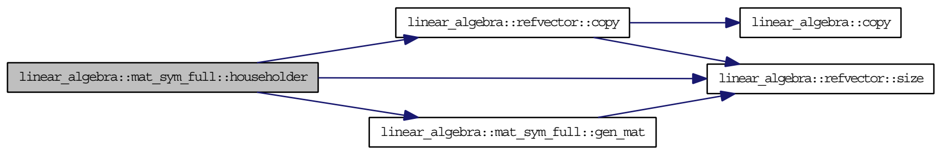
Here is the call graph for this function:



4.31.3.11 template<class TN> mat_full< TN > linear_algebra::mat_sym_full< TN >::householder (refvector< TN > & *diagonal*, refvector< TN > & *subdiagonal*)

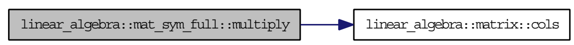
The aspired transformation is a product of $(I - v_i v_i^*)$ where v_i is $(\pm e_1 \|a_{12}\| + a_{12})/\sqrt{\pm a_{12,1}\|a_{12}\| + \|a_{12}\|^2}$. Notice that we begin counting at 0. The matrix is split up according to $A = \begin{pmatrix} a_{11} & a_{12}^* \\ a_{12} & A_{22} \end{pmatrix}$ We use preconditioning such that the matrix is ordered with lowest off-diagonal value in the low right=hand corner. $\begin{pmatrix} a_{11} & \cdots & & \\ & \ddots & & \\ & & \ddots & 0 \\ & & & a_{nn} \end{pmatrix}$ The returned matrix U will transfrom S to tridiagonal form via U^*SU.

Here is the call graph for this function:



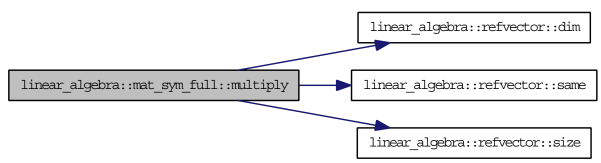
4.31.3.12 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::multiply (const TN & *a*, mat_sym_full< TN > & *r*) const

Here is the call graph for this function:



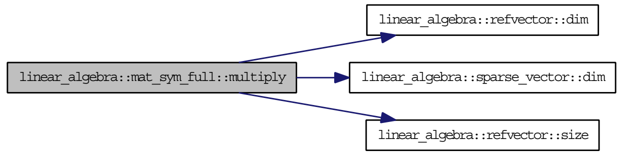
4.31.3.13 template<class TN> template<class TN2> refvector< TN > & linear_algebra::mat_sym_full< TN >::multiply (const refvector< TN2 > & *v*, refvector< TN > & *r*) const

Here is the call graph for this function:



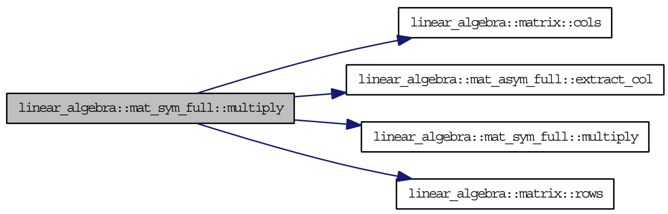
4.31.3.14 template<class TN> template<class TN2> refvector< TN > & linear_algebra::mat_sym_full< TN >::multiply (const sparse_vector< TN2 > & *v*, refvector< TN > & *r*) const

Here is the call graph for this function:



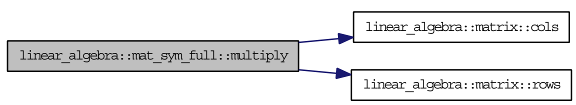
4.31.3.15 template<class TN> mat_full< TN > & linear_algebra::mat_sym_full< TN >::multiply (const mat_asym_full< TN > & *b*, mat_full< TN > & *r*) const

Here is the call graph for this function:



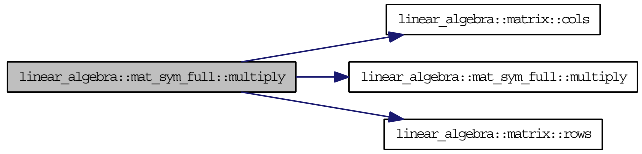
4.31.3.16 template<class TN> mat_full< TN > & linear_algebra::mat_sym_full< TN >::multiply (const mat_sym_full< TN > & *b*, mat_full< TN > & *r*) const

Here is the call graph for this function:



4.31.3.17 template<class TN> mat_full< TN > & linear_algebra::mat_sym_full< TN >::multiply (const mat_full< TN > & *b*, mat_full< TN > & *r*) const

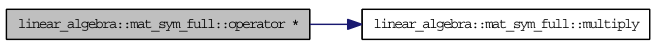
Here is the call graph for this function:



4.31.3.18 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::multiply (long *x*, long *y*, TN *a*)

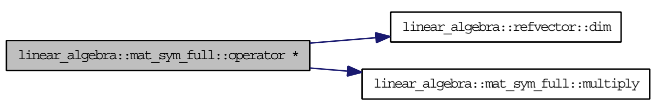
4.31.3.19 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::operator * (const TN & *a*) const

Here is the call graph for this function:



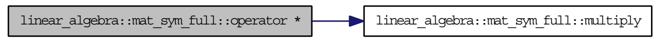
4.31.3.20 template<class TN> template<class TN2> refvector< TN > linear_algebra::mat_sym_full< TN >::operator * (const refvector< TN2 > & *v*) const

Here is the call graph for this function:



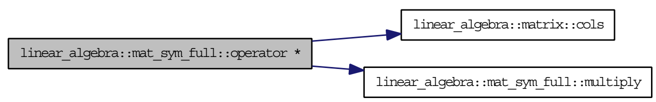
4.31.3.21 template<class TN> template<class TN2> refvector< TN >linear_algebra::mat_sym_full< TN >::operator * (const sparse_vector< TN2 > & *v*) const

Here is the call graph for this function:



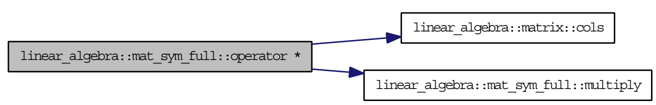
4.31.3.22 template<class TN> mat_full< TN > linear_algebra::mat_sym_full< TN >::operator ∗ (const mat_asym_full< TN > & *b*) const

Here is the call graph for this function:



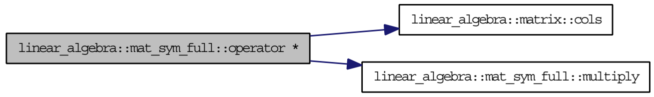
4.31.3.23 template<class TN> mat_full< TN > linear_algebra::mat_sym_full< TN >::operator ∗ (const mat_sym_full< TN > & *b*) const

Here is the call graph for this function:



4.31.3.24 template<class T> mat_full< T > linear_algebra::mat_sym_full< T >::operator ∗ (const mat_full< TN > & *b*) const

Here is the call graph for this function:



4.31.3.25 template<class TN> template<class TN2> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::operator *= (const TN2 & *a*)

4.31.3.26 template<class TN> refvector< TN > linear_algebra::mat_sym_full< TN >::operator() (const long *column*) const

4.31.3.27 template<class TN> refvector< TN > linear_algebra::mat_sym_full< TN >::operator() (const long *column*, const long *xmin*, const long *xmax*) const

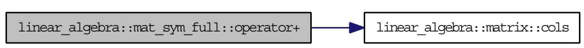
4.31.3.28 template<class TN> TN & linear_algebra::mat_sym_full< TN >::operator() (const long *x*, const long *y*)

4.31.3.29 template<class TN> const TN & linear_algebra::mat_sym_full< TN >::operator() (const long *x*, const long *y*) const[virtual]

Implements linear_algebra::matrix< TN >.

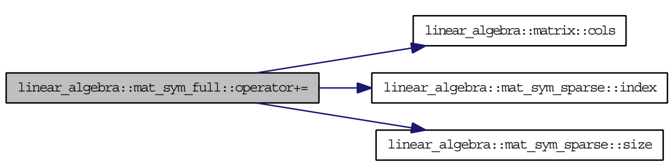
4.31.3.30 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::operator+ (const mat_sym_full< TN > & *a*) const

Here is the call graph for this function:



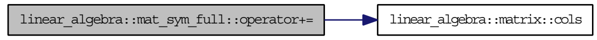
4.31.3.31 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::operator+= (const mat_sym_sparse< TN > & *a*)

Here is the call graph for this function:



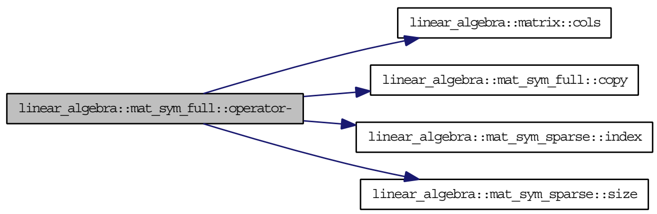
4.31.3.32 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::operator+= (const mat_sym_full< TN > & *a*)

Here is the call graph for this function:



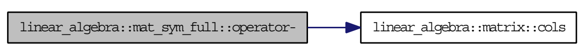
4.31.3.33 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::operator- (const mat_sym_sparse< TN > & *a*) const

Here is the call graph for this function:



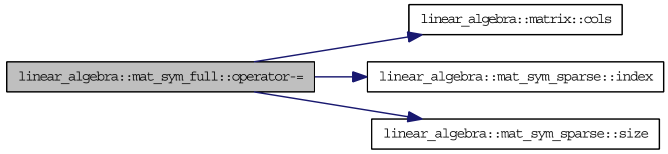
4.31.3.34 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::operator- (const mat_sym_full< TN > & *a*) const

Here is the call graph for this function:



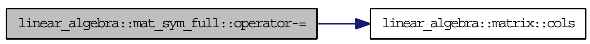
4.31.3.35 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::operator-= (const mat_sym_sparse < TN > & *a*)

Here is the call graph for this function:



4.31.3.36 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::operator-= (const mat_sym_full< TN > & *a*)

Here is the call graph for this function:



4.31.3.37 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::operator/ (const TN & *a*) const

4.31.3.38 template<class TN> template<class TN2> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::operator/= (const TN2 & *a*)

4.31.3.39 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::operator= (const mat_sym_full< TN > & *a*)

4.31.3.40 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::operator= (mat_sym_full< TN > & *a*)

The default is to output a full (column) matrix.

4.31.3.41 template<class TN> void linear_algebra::mat_sym_full< TN >::operator>> (ostream & *OUT*) const

The default is to output a full (column) matrix.

4.31.3.42 template<class TN> TN & linear_algebra::mat_sym_full< TN >::operator[] (const long)

4.31.3.43 template<class TN> const TN & linear_algebra::mat_sym_full< TN >::operator[] (const long *x*) const

4.31.3.44 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::prune (const TN *tol*)

Reimplemented from linear_algebra::symmetric< TN >.

4.31.3.45 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::resize (long *n*)

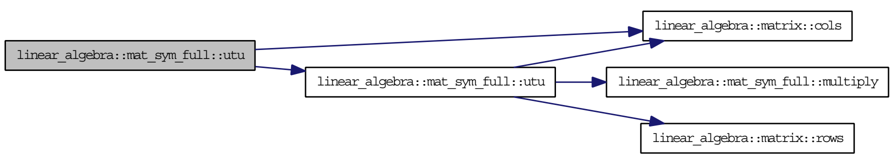
4.31.3.46 template<class TN> void linear_algebra::mat_sym_full< TN >::set (const long *x*, const long *y*, TN *value*)

4.31.3.47 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::unitary_transform (const mat_full< TN > *U*) const

This transform is only suggested if memory management is of importance. Otherwise using two matrix mulitplications is faster O(n^3).

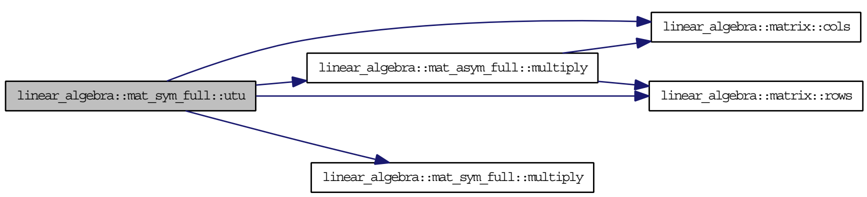
4.31.3.48 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::utu (const mat_asym_full< TN > & *U*) const

Here is the call graph for this function:



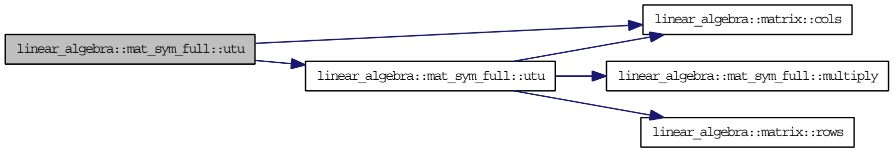
4.31.3.49 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::utu (const mat_asym_full< TN > & *a*, mat_full< TN > & *I*) const

Here is the call graph for this function:



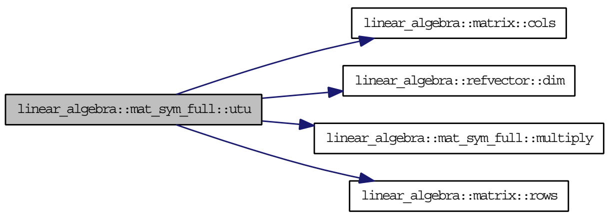
4.31.3.50 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::utu (const mat_full< TN > & *U*) const

Here is the call graph for this function:



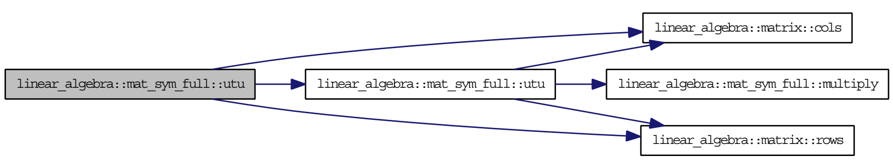
4.31.3.51 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::utu (const mat_full< TN > & U, mat_sym_full< TN > & R, refvector< TN > & v) const

Here is the call graph for this function:



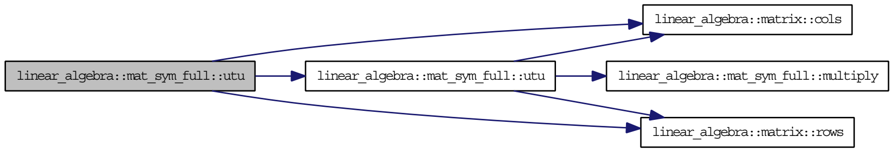
4.31.3.52 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::utu (const mat_full< TN > & U, mat_sym_full< TN > & R) const

Here is the call graph for this function:



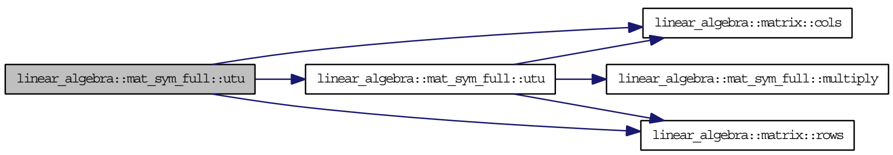
4.31.3.53 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::utu (const mat_sym_full< TN > & a) const

Here is the call graph for this function:



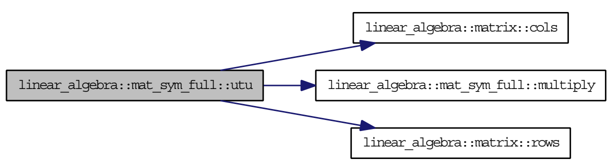
4.31.3.54 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::utu (const mat_sym_full< TN > & a, mat_sym_full< TN > &) const

Here is the call graph for this function:



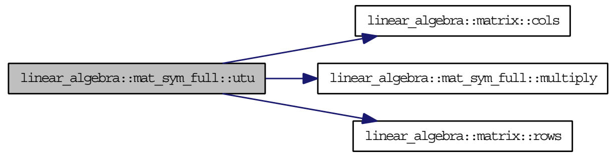
4.31.3.55 template<class TN>mat_sym_ful< TN > linear_algebra::mat_sym_full< TN >::utu (const mat_sym_full< TN > & a, mat_full< TN > & I) const

Here is the call graph for this function:



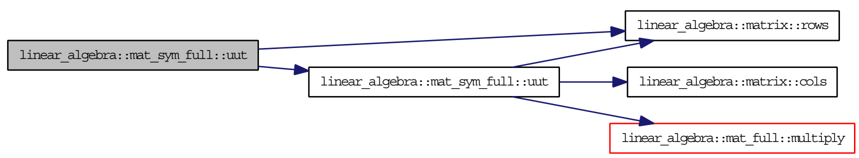
4.31.3.56 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::utu (const mat_sym_full< TN > & a, mat_sym_full< TN > & R, mat_full< TN > & I) const

Here is the call graph for this function:



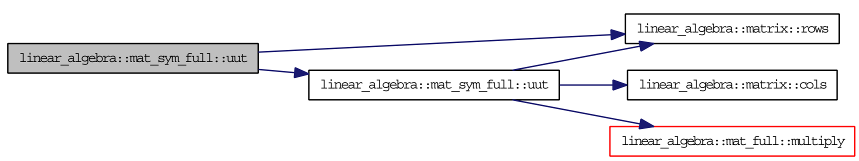
4.31.3.57 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::uut (const mat_full< TN > & U) const

Here is the call graph for this function:



4.31.3.58 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::uut (const mat_full< TN > & U, mat_sym_full< TN > & R) const

Here is the call graph for this function:



4.31.3.59 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::uut (const mat_full< TN > & U, mat_sym_full< TN > & R, mat_full< TN > & I) const

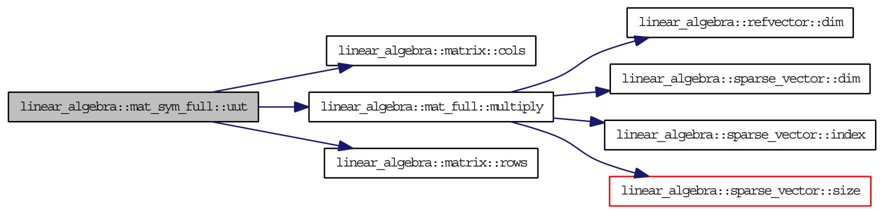
Here is the call graph for this function:

4.31.3.60 template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::zero ()

4.31.4 Member Data Documentation

4.31.4.1 template<class TN> refvector<TN> linear_algebra::mat_sym_full< TN >::_vals[private]

The documentation for this class was generated from the following files:

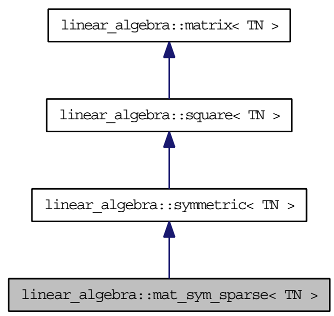


- include/BCR_CPP_LA/mat_sym_full.decl
- include/BCR_CPP_LA/mat_sym_full.h

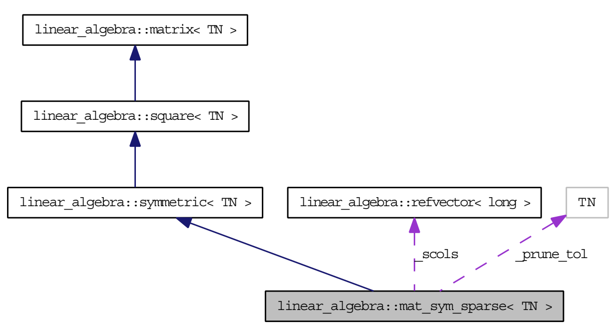
4.32 linear_algebra::mat_sym_sparse< TN > Class Template Reference

mat_sym_sparse packs sparse column matrices into a comfortable format.

Inheritance diagram for linear_algebra::mat_sym_sparse< TN >:



Collaboration diagram for linear_algebra::mat_sym_sparse< TN >:



Public Member Functions

- mat_sym_sparse ()
  Default constructor.
- mat_sym_sparse (long n)
  Constructor of dimension n with uninitialised values.
- mat_sym_sparse (long n, const vector< long > &scols, const vector< sparse_vector< TN > > &vals)
  Constructor of dimension n with initialisation of values.
- mat_sym_sparse (long n, const vector< long > &scols, refvector< sparse_vector< TN > > &vals)
  Constructor of dimension n with initialisation of values.
- mat_sym_sparse (long n, refvector< long > &scols, refvector< sparse_vector< TN > > &vals)
  Constructor of dimension n with initialisation of values.
- mat_sym_sparse (long n, refvector< long > &scols, const vector< sparse_vector< TN > > &vals)
  Constructor of dimension n with initialisation of values.
- mat_sym_sparse (istream &IN)
  Constructor to read a stored matrix from a stream. Aimed at retrieving information from a binary save with mat_sym_sparse::operator>>().
- mat_sym_sparse< TN > & operator= (mat_sym_sparse< TN > &b)
- mat_sym_sparse< TN > & operator= (const mat_sym_sparse< TN > &b)
- const TN & operator() (const long x, const long y) const
  Access operator.
- const sparse_vector< TN > operator[ ] (const long i) const
  Direct access operator.
- sparse_vector< TN > & operator[ ] (const long i)
  Direct access operator.
- void set (const long x, const long y, TN value)
  Set operation for safe assignments.
- mat_sym_sparse< TN > operator ∗ (const TN &a) const
  Multiplies a symmetric sparse matrix with an object.

- mat_full< TN > operator * (mat_full< TN > A) const
  Multiply by a full matrix.
- mat_full< TN > & multiply (mat_full< TN > A, mat_full< TN > &r) const
  Multiply by a full matrix.
- refvector< TN > & multiply (const refvector< TN > &v, refvector< TN > &r) const
  Multiply a symmetric sparse matrix with a full vector into a full vector./test.
- refvector< TN > operator * (const refvector< TN > &v) const
  Multiply a symmetric sparse matrix with a full vector into a full vector./test.
- mat_sym_sparse< TN > & operator *= (const TN &x)
  Multiply by an object of TN into left side.
- mat_sym_sparse< TN > & multiply (long x, long y, TN a)
  Multiplies an element by a number.
- mat_full< TN > & multiply (const mat_asym_full< TN > &B, mat_full< TN > &r) const
  Multiply a symmetric sparse matrix with a full vector into a full vector./test.
- mat_full< TN > operator * (const mat_asym_full< TN > &B) const
  Multiply a symmetric sparse matrix with a full vector into a full vector./test.
- mat_sym_sparse & operator/= (const TN &x)
  Divide by an object of TN into left side. This is very slow and not advisable.
- sparse_vector< TN > operator * (const sparse_vector< TN > &v) const
  Multiply a sparse symmetric matrix with a sparse vector.
- mat_sparse< TN > operator * (const mat_sparse< TN > &B) const
  Multiply a sparse symmetric matrix with a sparse matrix.
- mat_sym_sparse< TN > & operator+= (const mat_sym_sparse< TN > &B)
  Add a matrix into left object.
- mat_sym|_sparse< TN > & operator-= (const mat_sym_sparse< TN > &B)
  Subtract a matrix into left object.
- bool is_col_nonzero (const long x, long &i) const
- bool is_nonzero (long x, long y, long &j, long &k) const
  Checks whether an access is non zero.
- bool add (long x, long y, const TN z)
  Adds a value.

- bool is_empty () const
- long size () const
  Returns size of non-zero entries in _svals.
- void operator>> (ostream &OUT) const
  Output matrix in binary format into a stream. Aimed at file storage.
- mat_sym_sparse< TN > & prune (const TN x=0)
  Deletes all entries that are smaller in magnitude than x.
- TN max_element () const
  Finds the maximum element.
- bool display () const
  Simple display of matrix.
- vector< TN > extract_full_col (const long column, const long xmin, const long xmax) const
  Extracts a block from a column/row.
- vector< TN > extract_full_col (const long column, const long xmin=0) const
  Extracts a block from a column/row.
- mat_sym_full< TN > extract_full_sym_block (long xmin, long xmax) const
  Extracts a full symmetric block.
- mat_sym_full< TN > extract_full_sym_block (long xmin=0) const
  Extracts a full symmetric block.
- mat_sparse< TN > extract_sparse_block (long xmin, long xmax, long ymin, long ymax) const
  Extracts a block from anywhere in the matrix.
- sparse_vecto< TN > extract_sparse_col (const long column, const long xmin=0) const
  Extracts a block from a column/row.
- sparse_vector< TN > extract_sparse_lower_col (const long column, const long xmin, const long xmax) const
  Extracts a sparse column up to the diagonal.
- sparse_vector< TN > extract_sparse_lower_col (const long column, const long xmin=0) const
  Extracts a block from a column/row.

- double density () const
  Return the density of non-zero elements.

- void gen_mat (const sparse_vector< TN > &a, const sparse_vector< TN > &b, TN correction)
  Produce the necessary $ij^* + ji^*$.

- mat_sparse< TN > householder (vector< TN > &diagonal, vector< TN > &subdiagonal)
  This routine produces a the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.

- mat_sparse< TN > householder (refvector< TN > &diagonal,refvector< TN > &subdiagonal)
  This routine produces a the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.

- void clear ()
  Sets matrix to zero releasing all associated memory.

- mat_sym_sparse< TN > & zero ()
  Sets matrix to zero releasing all associated memory.

- long index (long i) const
  Returns the ith nonzero column number.

- mat_sym_full< TN > utu (const mat_full< TN > &U, mat_sym_full< TN > &R) const
  This is the unitary transform U^*AU.

- mat_sym_full< TN > utu (const mat_full< TN > &U) const
  This is the unitary transform U^tAU.

- TN trace () const
  The square matrix class contains all n by n matrices.

- void copy (mat_sym_sparse< TN > &b, long i)

- void copy (const mat_sym_sparse< TN > &b)

- void copy (mat_sym_sparse< TN > &b)

- symmetric< TN > & copy (const symmetric< TN > &a)
  Copy matrix with depth 1.

- symmetric< TN > & copy (const symmetric< TN > &a, const long i)
  Copy matrix with depth i.

- symmetric< TN > & copy (symmetric< TN > &a, const long i)
  Copy matrix with depth i.

Public Attributes

- TN _prune_tol
  Tolerance for multiplication with a sparse vector.

Private Attributes

- refvector< long > _scols
  Vector of column indices.

- refvector< sparse_vector< TN > > _svals
  Vector of sparse columns.

- bool _pt
  State of matrix.

Friends

- class mat_sparse< TN >
- class mat_full< TN >
- class mat_asym_full< TN >

4.32.1 Detailed Description

template<class TN> class linear_algebra::mat_sym_sparse< TN >

The sparse matrix is packed as a sparse vector of sparse vectors. Only the columns which are not empty are indexed and kept in memory. The sparse vector part takes care of the rows.

4.32.2 Constructor & Destructor Documentation

4.32.2.1 template<class TN> linear_algebra::mat_sym_sparse< TN >::mat_sym_sparse ()

4.32.2.2 template<class TN> linear_algebra::mat_sym_sparse< TN >::mat_sym_sparse (long *n*)

4.32.2.3 template<class TN> linear_algebra::mat_sym_sparse< TN >::mat_sym_sparse (long *n*, const vector< long > & *scols*, const vector< sparse_vector< TN > > & *vals*)

Parameters:

n Dimension of matrix.
srows Vector of row indices.
scols Vector of column indices.
vals Vector of values.

Each value has an associated row and column index in srows and scols. Therefore they all have the same size. There is no integrity check.

4.32.2.4 template<class TN> linear_algebra::mat_sym_sparse< TN >::mat_sym_sparse (long *n*, const vector< long > & *scols*, refvector< sparse_vector< TN > > & *vals*)

Parameters:

n Dimension of matrix.
srows Vector of row indices.
scols Vector of column indices.
vals reference counted Vector of values.

Each value has an associated row and column index in srows and scols. Therefore they all have the same size. There is no integrity check.

4.32.2.5 template<class TN> linear_algebra::mat_sym_sparse< TN >::mat_sym_sparse (long *n*, refvector< long > & *scols*, refvector< sparse_vector< TN > > & *vals*)

Parameters:

n Dimension of matrix.
srows Vector of row indices.
scols Reference counted vector of column indices.
vals Reference counted vector of values.

Each value has an associated row and column index in srows and scols. Therefore they all

have the same size. There is no integrity check.

Here is the call graph for this function:

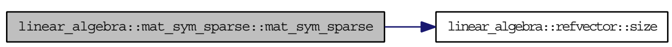
4.32.2.6 template<class TN> linear_algebra::mat_sym_sparse< TN >::mat_sym_sparse (long *n*, refvector< long > & *scols*, const vector< sparse_vector< TN > > & *vals*)

Parameters:

n Dimension of matrix.
srows Vector of row indices.
scols Reference counted vector of column indices.
vals Vector of values.

Each value has an associated row and column index in srows and scols. Therefore they all have the same size. There is no integrity check.

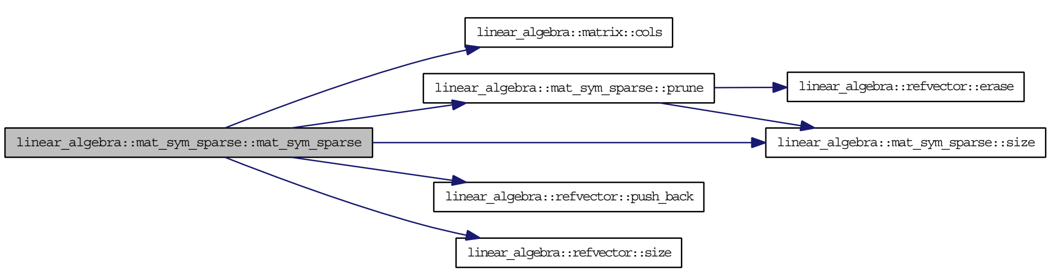
Here is the call graph for this function:



4.32.2.7 template<class TN> linear_algebra::mat_sym_sparse< TN >::mat_sym_sparse (istream & *IN*)

The resultant vector is pruned to default pruning tolerance = 0. This should be completely superfluous but in case a different program not using this class produces the matrix as input we prune.

Here is the call graph for this function:



4.32.3 Member Function Documentation

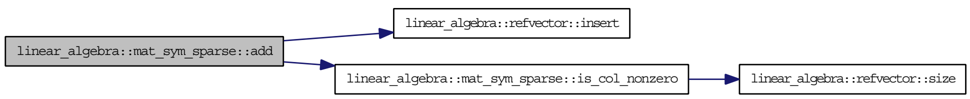
4.32.3.1 template<class TN> bool linear_algebra::mat_sym_sparse< TN >::add (long x, long y, const TN z)

Parameters:

> z is added to (x,y). If the value is zero the associated vectors are expanded.

Reimplemented from linear_algebra::symmetric< TN >.

Here is the call graph for this function:



4.32.3.2 template<class TN> void linear_algebra::mat_sym_sparse< TN >::clear ()

4.32.3.3 template<class TN>symmetric< TN > & linear_algebra::mat_sym_sparse< TN >::copy (symmetric< TN > & a, const long i)[virtual]

Implements linear_algebra::symmetric< TN >.

4.32.3.4 template<class TN> symmetric< TN > & linear_algebra::mat_sym_sparse< TN >::copy (const symmetric< TN > & a, const long i)[virtual]

Implements linear_algebra::symmetric< TN >.

4.32.3.5 template<class TN> symmetric< TN > & linear_algebra::mat_sym_sparse< TN >::copy (const symmetric< TN > & a)[virtual]

Implements linear_algebra::symmetric< TN >.

4.32.3.6 template<class TN> void linear_algebra::mat_sym_sparse< TN >::copy (mat_sym_sparse< TN > & b)

4.32.3.7 template<class TN> void linear_algebra::mat_sym_sparse< TN >::copy (const mat_sym_sparse< TN > & b)

4.32.3.8 template<class TN> void linear_algebra::mat_sym_sparse< TN >::copy (mat_sym_sparse< TN > & b, long i)

4.32.3.9 template<class TN> double linear_algebra::mat_sym_sparse< TN >::density () const

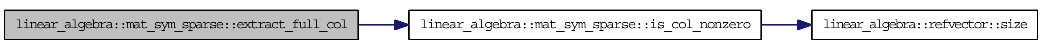
4.32.3.10 template<class TN> bool linear_algebra::mat_sym_sparse< TN >::display () const

4.32.3.11 template<class TN> vector< TN > linear_algebra::mat_sym_sparse< TN >::extract_full_col (const long *column*, const long *xmin* = 0) const

Here is the call graph for this function:

4.32.3.12 template<class TN> vector< TN > linear_algebra::mat_sym_sparse< TN >::extract_full_col (const long *column*, const long *xmin*, const long *xmax*) const

Here is the call graph for this function:

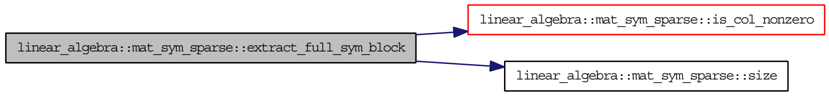


4.32.3.13 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_sparse< TN >::extract_full_sym_block (long *xmin* = 0) const

Here is the call graph for this function:

4.32.3.14 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_sparse< TN >::extract_full_sym_block (long *xmin*, long *xmax*) const

Here is the call graph for this function:

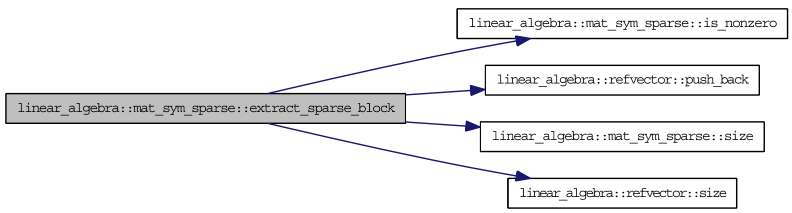


4.32.3.15 template<class TN> mat_sparse< TN > linear_algebra::mat_sym_sparse< TN >::extract_sparse_block (long *xmin*, long *xmax*, long *ymin*, long *ymax*) const

Test

should work, but ...

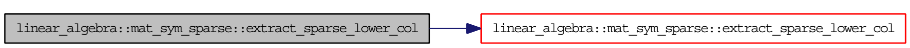
Here is the call graph for this function:



4.32.3.16 template<class TN> sparse_vector< TN > linear_algebra::mat_sym_sparse< TN >::extract_sparse_col (const long *column*, const long *xmin* = 0) const

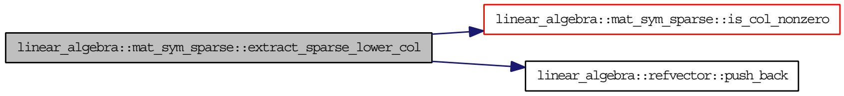
4.32.3.17 template<class TN> sparse_vector< TN > linear_algebra::mat_sym_sparse< TN >::extract_sparse_lower_col (const long *column*, const long *xmin* = 0) const

Here is the call graph for this function:



4.32.3.18 template<class TN> sparse_vector< TN > linear_algebra::mat_sym_sparse< TN >::extract_sparse_lower_col (const long *column*, const long *xmin*, const long *xmax*) const

Here is the call graph for this function:



4.32.3.19 template<class TN> long linear_algebra::mat_sym_sparse< TN >::index (long *i*) const

4.32.3.20 template<class TN> bool linear_algebra::mat_sym_sparse< TN >::is_col_nonzero (const long *x*, long & *i*) const

Here is the call graph for this function:

4.32.3.21 template<class TN> bool linear_algebra::mat_sym_sparse< TN >::is_empty () const

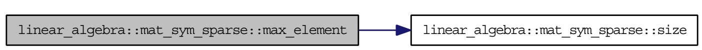
4.32.3.22 template<class TN> bool linear_algebra::mat_sym_sparse< TN >::is_nonzero (long *x*, long *y*, long & *j*, long & *k*) const

Parameters:

j holds the position of (x,y) in _vals. If (x,y) is 0. An insertion would happen here.

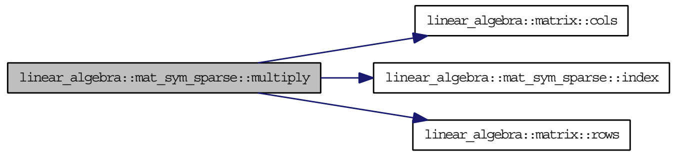
4.32.3.23 template<class TN> TN linear_algebra::mat_sym_sparse< TN >::max_element () const

Here is the call graph for this function:



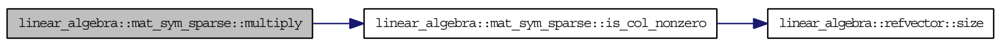
4.32.3.24 template<class TN> mat_full< TN > & linear_algebra::mat_sym_sparse< TN >::multiply (const mat_asym_full< TN > & *B*, mat_full< TN > & const

Here is the call graph for this function:



4.32.3.25 template<class TN> mat_sym_sparse< TN > & linear_algebra::mat_sym_sparse< TN >::multiply (long x, long y, TN a)

Here is the call graph for this function:



4.32.3.26 template<class TN> refvector< TN > & linear_algebra::mat_sym_sparse< TN >::multiply (const refvector< TN > & v, refvector< TN > & r) const

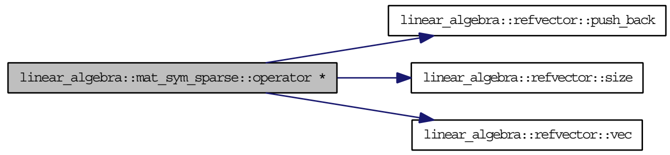
Here is the call graph for this function:

4.32.3.27 template<class TN> mat_full< TN > &linear_algebra::mat_sym_sparse< TN >::multiply (mat_full< TN > A, mat_full< TN > & r) const

Test

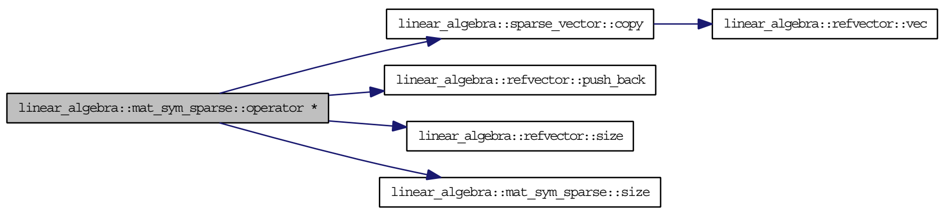
4.32.3.28 template<class TN> mat_sparse< TN > linear_algebra::mat_sym_sparse< TN >::operator $*$ (const mat_sparse< TN > & B) const

Here is the call graph for this function:



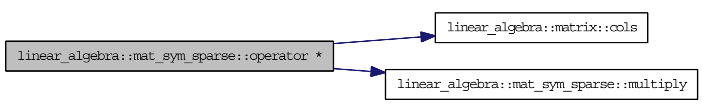
4.32.3.29 template<class TN> sparse_vector< TN > linear_algebra::mat_sym_sparse< TN >::operator * (const sparse_vector< TN > & *v*) const

Here is the call graph for this function:



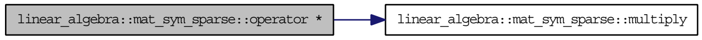
4.32.3.30 template<class TN> mat_full< TN > linear_algebra::mat_sym_sparse< TN >::operator * (const mat_asym_full< TN > & *B*) const

Here is the call graph for this function:



4.32.3.31 template<class TN> refvector< TN > linear_algebra::mat_sym_sparse< TN >::operator * (const refvector< TN > & *v*) const

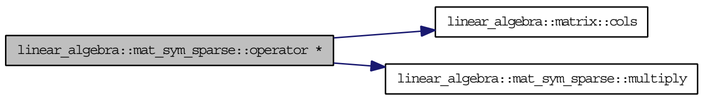
Here is the call graph for this function:



4.32.3.32 template<class TN> mat_full< TN > linear_algebra::mat_sym_sparse< TN >::operator $*$ (mat_full< TN > A) const

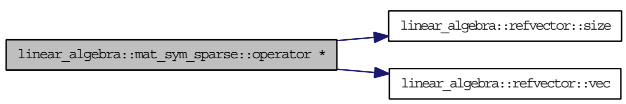
Test

Here is the call graph for this function:



4.32.3.33 template<class TN> mat_sym_sparse< TN > linear_algebra::mat_sym_sparse< TN >::operator $*$ (const TN & a) const

Here is the call graph for this function:



4.32.3.34 template<class TN> mat_sym_sparse< TN > & linear_algebra::mat_sym_sparse< TN >::operator $*=$ (const TN & x)

4.32.3.35 template<class TN> const TN & linear_algebra::mat_sym_sparse< TN >::operator() (const long x, const long y) const[virtual]

Implements linear_algebra::matrix< TN >.

Here is the call graph for this function:

4.32.3.36 template<class TN> mat_sym_sparse<TN>& linear_algebra::mat_sym_sparse< TN >::operator+= (const mat_sym_sparse< TN > & *B*)

4.32.3.37 template<class TN> mat_sym_sparse<TN>& linear_algebra::mat_sym_sparse< TN >::operator-= (const mat_sym_sparse< TN > & *B*)

4.32.3.38 template<class TN> mat_sym_sparse< TN > & linear_algebra::mat_sym_sparse< TN >::operator/= (const TN & *x*)

4.32.3.39 template<class TN> mat_sym_sparse< TN > & linear_algebra::mat_sym_sparse< TN >::operator= (const mat_sym_sparse< TN > & *b*)

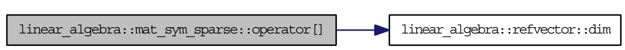
4.32.3.40 template<class TN> mat_sym_sparse< TN > & linear_algebra::mat_sym_sparse< TN >::operator= (mat_sym_sparse< TN > & *b*)

4.32.3.41 template<class TN> void linear_algebra::mat_sym_sparse< TN >::operator>> (ostream & *OUT*) const

Here is the call graph for this function:

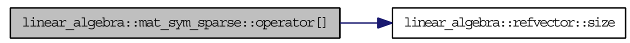
4.32.3.42 template<class TN> sparse_vector< TN > & linear_algebra::mat_sym_sparse< TN >::operator[] (const long *i*)

Here is the call graph for this function:



4.32.3.43 template<class TN> const sparse_vector< TN > linear_algebra::mat_sym_sparse< TN >::operator[] (const long *i*) const

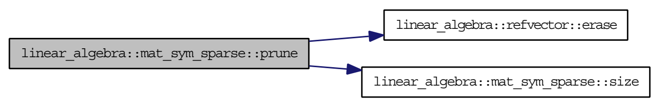
Here is the call graph for this function:



4.32.3.44 template<class TN> mat_sym_sparse< TN > & linear_algebra::mat_sym_sparse< TN >::prune (const TN $x = 0$)

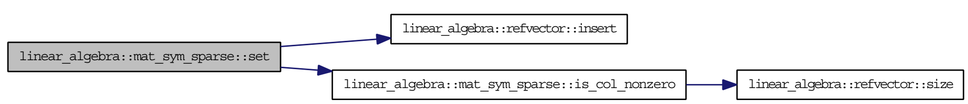
Reimplemented from linear_algebra::symmetric< TN >.

Here is the call graph for this function:



4.32.3.45 template<class TN> void linear_algebra::mat_sym_sparse< TN >::set (const long x, const long y, TN *value*)

Here is the call graph for this function:



4.32.3.46 template<class TN> long linear_algebra::mat_sym_sparse< TN >::size () const

4.32.3.47 template<class TN> TN linear_algebra::mat_sym_sparse< TN >::trace () const

Reimplemented from linear_algebra::square< TN >.

4.32.3.48 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_sparse< TN >::utu (const mat_full< TN > & U) const

4.32.3.49 template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_sparse< TN >::utu (const mat_full< TN > & U, mat_sym_full< TN > & R) const

4.32.3.50 template<class TN> mat_sym_sparse< TN > & linear_algebra::mat_sym_sparse< TN >::zero ()

4.32.4 Friends and Related Function Documentation

4.32.4.1 template<class TN> friend class mat_asym_full< TN >[friend]

4.32.4.2 template<class TN> friend class mat_full< TN >[friend]

4.32.4.3 template<class TN> friend class mat_sparse< TN >[friend]

4.32.5 Member Data Documentation

4.32.5.1 template<class TN> bool linear_algebra::mat_sym_sparse< TN >::_empty[private]

4.32.5.2 template<class TN> TN linear_algebra::mat_sym_sparse< TN >::_prune_tol

4.32.5.3 template<class TN> refvector<long> linear_algebra::mat_sym_sparse< TN >::_scols[private]

4.32.5.4 template<class TN> refvector<sparse_vector<TN> > linear_algebra::mat_sym_sparse< TN >::_svals[private]

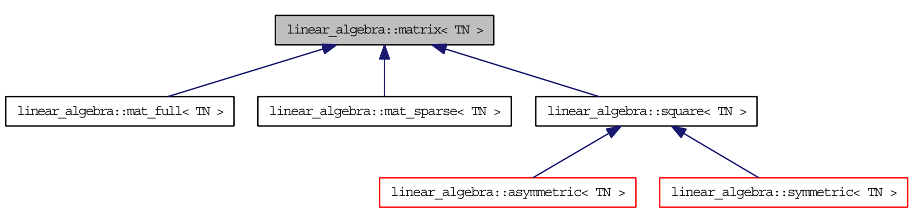
The documentation for this class was generated from the following files:

- include/BCR_CPP_LA/mat_sym_sparse.decl
- include/BCR_CPP_LA/mat_sym_sparse.h

4.33 linear_algebra::matrix< TN > Class Template Reference

matrix is a basic class which covers all kinds of matrices.

Inheritance diagram for linear_algebra::matrix< TN >:



Public Member Functions

- matrix ()
  Default constructor.
- virtual ~matrix ()
  Default destructor.

- virtual void operator>> (std::ostream &OUT) const
  Output matrix in binary format into a stream. Aimed at file storage.
- virtual const TN & operator() (const long i, const long j) const =0
- long rows () const
  Returns number of rows.
- long cols () const
  Returns number of columns.
- long size ()
  Returns the number of entries.
- matrix< TN > & operator= (const matrix< TN > &b)
- TN trace () const
  We demand that there is a zero function.

Protected Attributes

- long _rows
  Number of rows.
- long _cols
  Number of columns.
- long _size
  Size of std::vector<TN> _vals.

template<class TN> class linear_algebra::matrix< TN >

4.33.1 Constructor & Destructor Documentation

4.33.1.1 template<class TN> linear_algebra::matrix< TN >::matrix ()

matrix is a basic class which covers all kinds of matrices. Default constructor.

4.33.1.2 template<class TN> virtual linear_algebra::matrix< TN >::~matrix ()[inline, virtual]

4.33.2 Member Function Documentation

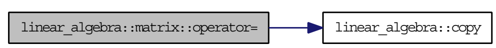
4.33.2.1 template<class TN> long linear_algebra::matrix< TN >::cols () const

4.33.2.2 template<class TN> virtual const TN& linear_algebra::matrix< TN >::operator() (const long *i*, const long *j*) const[pure virtual]

Implemented in linear_algebra::mat_asym_full< TN >, linear_algebra::mat_asym_sparse< TN >, linear_algebra::mat_full< TN >, linear_algebra::mat_sparse< TN >,linear_algebra::mat_sym_full< TN >, linear_algebra::mat_sym_sparse< TN >, linear_algebra::mat_full< double >, linear_algebra::mat_full< bool >, and linear_algebra::mat_sym_full< valerg >.

4.33.2.3 template<class TN> matrix< TN > & linear_algebra::matrix< TN >::operator= (const matrix< TN > & *b*)

Here is the call graph for this function:



4.33.2.4 template<class TN> virtual void linear_algebra::matrix< TN >::operator>> (std::ostream & *OUT*) const[inline, virtual]

The default is to output a full (column) matrix.

Reimplemented in linear_algebra::mat_asym_full< TN >.

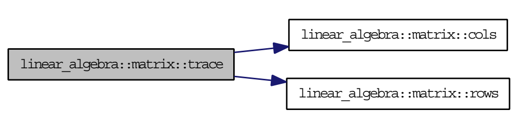
4.33.2.5 template<class TN> long linear_algebra::matrix< TN >::rows () const

4.33.2.6 template<class TN> long linear_algebra::matrix< TN >::size ()

4.33.2.7 template<class TN> TN linear_algebra::matrix< TN >::trace () const

Reimplemented in linear_algebra::mat_sym_sparse< TN >, linear_algebra::square< TN >, and linear_algebra::square< valerg >.

Here is the call graph for this function:



4.33.3 Member Data Documentation

4.33.3.1 template<class TN> long linear_algebra::matrix< TN >::_cols[protected]

4.33.3.2 template<class TN> long linear_algebra::matrix< TN >::_rows[protected]

4.33.3.3 template<class TN> long linear_algebra::matrix< TN >::_size[protected]

The documentation for this class was generated from the following files:

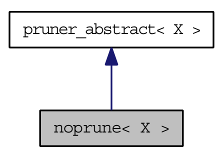
- include/BCR_CPP_LA/matrix.decl
- include/BCR_CPP_LA/matrix.h

4.34 noprune< X > Class Template Reference

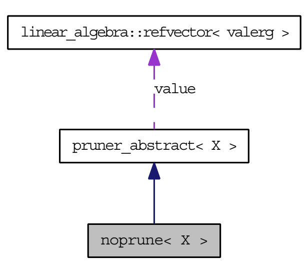
Dummy pruning class. No actual pruning done.

#include <noprune.h>

Inheritance diagram for noprune< X >:



Collaboration diagram for noprune< X >:



Public Member Functions

- noprune ()

- noprune (const noprune< X > &a)
- void adjust_lagrange (double &lambda, ulong &conf1, ulong &conf2, long &config, const refvector< ulong > &visited_run) const
  Compute the increase of lambda and assess current best value.
- ulong prune (double &lambda, ulong &conf1, ulong &conf2, long &config, const refvector< ulong > &a) const
- ulong deprune (ulong N const
  self-explanatory
- ulong reprune (ulong N) const
  Given an "absolute" reference number return the current reference number.

template<class X> class noprune< X >

4.34.1 Constructor & Destructor Documentation

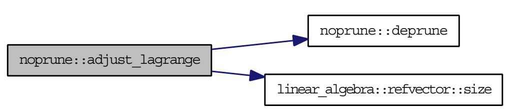
4.34.1.1 template<class X> noprune< X >::noprune ()[inline]

4.34.1.2 template<class X> noprune< X >::noprune (const noprune< X > & *a*)[inline]

4.34.2 Member Function Documentation

4.34.2.1 template<class X> void noprune< X >::adjust_lagrange (double & *lambda*, ulong & *conf1*, ulong & *conf2*, long & *config*, const refvector< ulong > & *visited_run*) const[inline]

Here is the call graph for this function:



4.34.2.2 template<class X> ulong noprune< X >::deprune (ulong *N*) const[inline, virtual]

Implements pruner_abstract< X >.

4.34.2.3 template<class X> ulong noprune< X >::prune (double & *lambda*, ulong & *conf1*, ulong & *conf2*, long & *config*, const refvector< ulong > & *a*) const[inline, virtual]

Implements pruner_abstract< X >.

Here is the call graph for this function:

4.34.2.4 template<class X> ulong noprune< X >::reprune (ulong N) const[inline, virtual]

Implements pruner_abstract< X >.

The documentation for this class was generated from the following file:

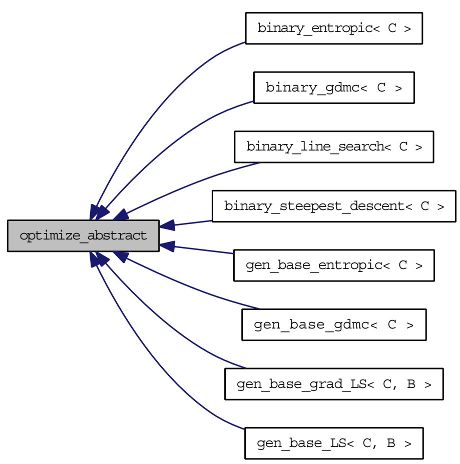
- noprune.h

4.35 optimize_abstract Class Reference

Abstract class that implements the Concepts::has_optimize concept. Only good for generic referncing of optimizations.

#include <optimizeabstract.h>

Inheritance diagram for optimize_abstract:



Public Member Functions

- virtual ~optimize_abstract ()

- ulong optimize () const
  Default optimize.
- virtual ulong optimize (const ulong N) const =0
- optimize_abstract ()
- optimize_abstract (const optimize_abstract &a)
- void set_id (const string &ID) const

Public Attributes

- const string & id_r

Private Attributes

- string id
  Identifier string;.

4.35.1 Constructor & Destructor Documentation

4.35.1.1 virtual optimize_abstract::∼optimize_abstract ()text[inline, virtual]

4.35.1.2 optimize_abstract::optimize_abstract ()[inline]

4.35.1.3 optimize_abstract::optimize_abstract (const optimize_abstract & *a*)[inline]

4.35.2 Member Function Documentation

4.35.2.1 virtual ulong optimize_abstract::optimize (const ulong *N*) const[pure virtual]

Implemented in binary_entropic< C >, binary_line_search< C >, binary_gdmc< C >, binary_steepest_descent< C >, gen_base_entropic< C >, gen_base_grad_LS< C, B >, gen_base_LS< C, B >, and gen_base_gdmc< C >.

4.35.2.2 ulong optimize_abstract::optimize () const[inline]

4.35.2.3 void optimize_abstract::set_id (const string & *ID*) const[inline]

4.35.3 Member Data Documentation

4.35.3.1 string optimize_abstract::id[mutable, private]

4.35.3.2 const string& optimize_abstract::id_r

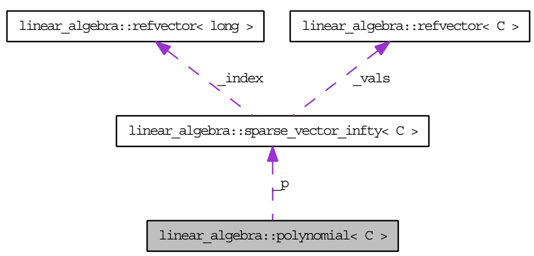
The documentation for this class was generated from the following file:

- optimizeabstract.h

4.36 linear_algebra::polynomial< C > Class Template Reference

A class for polynomials.

Collaboration diagram for linear_algebra::polynomial< C >:



Public Member Functions

- polynomial ()
  Standard constructor.
- polynomial< C > & operator= (const polynomial< C > &a)
  Assignment operator.
- polynomial< C > & operator= (polynomial< C > &a)
  Assignment operator.
- polynomial< C > & zero ()
  Define a zero.
- long index (const long i) const
  Get the exponent of the ith element.

- long size () const
  return the number of nonzeros.

Multiplications

- polynomial< C > & multiply (const polynomial< C > &a, polynomial< C > &r) const
  Multiply two polynomials.
- polynomial< C > operator * (const polynomial< C > &a) const
  Multiply two polynomials.
- polynomial< C > operator *= (const polynomial< C > &a)
  Multiply two polynomials.

Additions

- polynomial< C > operator+ (const polynomial< C > &a) const
  Add two polynomials.
- polynomial< C > & operator+= (const polynomial< C > &a)
  Add two polynomials.

Access

- const C & operator() (const long i) const
  Acess an element.
- C & operator() (const long i)
  Acess an element.

Initialisers

- void set (const long i, const C &a)
  The ith element to a.
- void set (const long i, C &a)
  The ith element to a.

Direct Acces operators

- C operator[ ] (const long i) const
- C & operator[ ] (const long i)

Copy operations.

- polynomial< C > & copy (const polynomial< C > &a)
  Copy assignment.
- polynomial< C > & copy (polynomial< C > &a)
  Copy assignment.
- polynomial< C > & copy (const polynomial< C > &a, const long i)
  Copy assignment.
- polynomial< C > & copy (polynomial< C > &a, const long i)
  Copy assignment.

Private Attributes

- sparse_vector_inft< C > _

template<class C> class linear_algebra::polynomial< C >

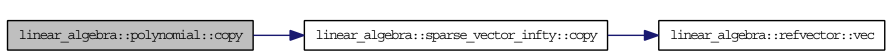
4.36.1 Constructor & Destructor Documentation

4.36.1.1 template<class C> linear_algebra::polynomial< C >::polynomial ()

4.36.2 Member Function Documentation

4.36.2.1 template<class C> polynomial< C > & linear_algebra::polynomial< C >::copy (polynomial< C > & *a*, const long *i*)

Here is the call graph for this function:



4.36.2.2 template<class C> polynomial< C > & linear_algebra::polynomial< C >::copy (const polynomial< C > & *a*, const long *i*)

Here is the call graph for this function:

4.36.2.3 template<class C> polynomial< C > & linear_algebra::polynomial< C >::copy (polynomial< C > & *a*)

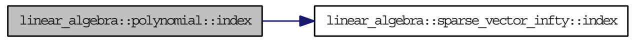
Here is the call graph for this function:

436.2.4 template<class C> polynomial< C > & linear_algebra::polynomial< C >::copy (const polynomial< C > & *a*)

Here is the call graph for this function:

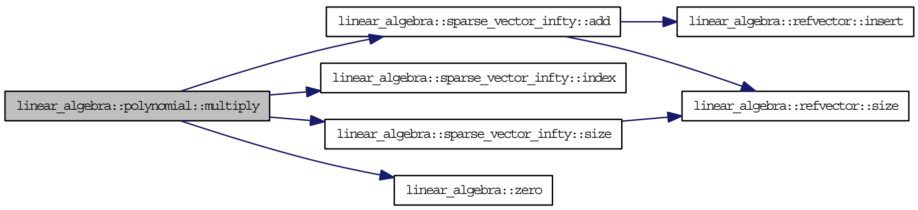
4.36.2.5 template<class C> long int linear_algebra::polynomial< C >::index (const long *i*) const

Here is the call graph for this function:



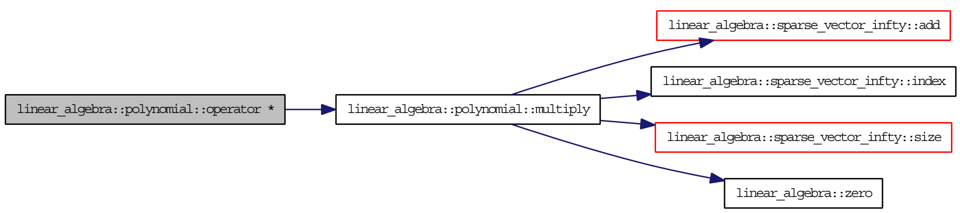
4.36.2.6 template<class C> polynomial< C > & linear_algebra::polynomial< C >::multiply (const polynomial< C > & *a*, polynomial< C > & *r*) const

Here is the call graph for this function:



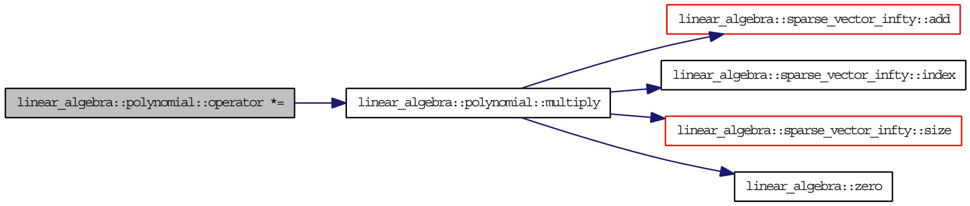
4.36.2.7 template<class C> polynomial< C > linear_algebra::polynomial< C >::operator * (const polynomial< C > & *a*) const

Here is the call graph for this function:



4.36.2.8 template<class C> polynomial< C > linear_algebra::polynomial< C >::operator *= (const polynomial< C > & *a*)

Here is the call graph for this function:

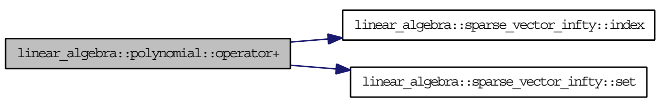


4.36.2.9 template<class C> C & linear_algebra::polynomial< C >::operator() (const long *i*)

4.36.2.10 template<class C> const C & linear_algebra::polynomial< C >::operator() (const long *i*) const

4.36.2.11 template<class C> polynomial< C > linear_algebra::polynomial< C >::operator+ (const polynomial< C > & *a*) const

Here is the call graph for this function:



4.36.2.12 template<class C> polynomial< C > & linear_algebra::polynomial< C >::operator+= (const polynomial< C > & *a*)

4.36.2.13 template<class C> polynomial< C > & linear_algebra::polynomial< C >::operator= (polynomial< C > & *a*)

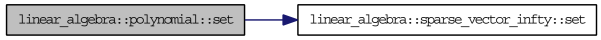
4.36.2.14 template<class C> polynomial< C > & linear_algebra::polynomial< C >::operator= (const polynomial< C > & *a*)

4.36.2.15 template<class C> C & linear_algebra::polynomial< C >::operator[] (const long *i*)

4.36.2.16 template<class C> C linear_algebra::polynomial< C >::operator[] (const long *i*) const

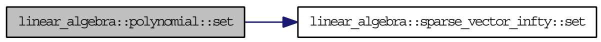
4.36.2.17 template<class C> void linear_algebra::polynomial< C >::set (const long *i*, C & *a*)

Here is the call graph for this function:



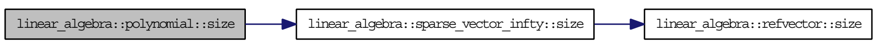
4.36.2.18 template<class C> void linear_algebra::polynomial< C >::set (const long *i*, const C & *a*)

Here is the call graph for this function:



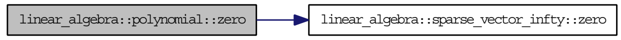
4.36.2.19 template<class C> long int linear_algebra::polynomia< C >::size () const

Here is the call graph for this function:



4.36.2.20 template<class C> polynomial< C > & linear_algebra::polynomial< C >::zero ()

Here is the call graph for this function:



4.36.3 Member Data Documentation

4.36.3.1 template<class C> sparse_vector_infty<C> linear_algebra::polynomial< C >::_p[private]

The documentation for this class was generated from the following files:

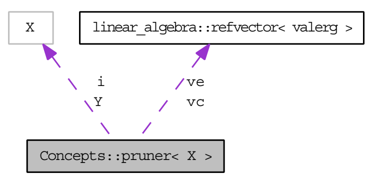
- include/BCR_CPP_LA/polynomial.decl
- include/BCR_CPP_LA/polynomial.h

4.37 Concepts::pruner< X > Class Template Reference

Concept defines an addressable set of values.

#include <concepts.hh>

Collaboration diagram for Concepts::pruner< X >:



Public Member Functions

- BOOST_CONCEPT_USAGE (pruner)

Private Member Functions

- BOOST_CONCEPT_ASSERT ((Library< X >))

Private Attributes

- X & i

- const X & Y
- refvector< ulong > v
- refvector< valerg >ve
- const refvector< ulong > vd
- const refvector< valerg > vc

template<class X> class Concepts::pruner< X >

4.37.1 Member Function Documentation

4.37.1.1 template<class X> Concepts::pruner< X >::BOOST_CONCEPT_ASSERT ((Library< X >))[private]

4.37.1.2 template<class X> Concepts::pruner< X >::BOOST_CONCEPT_USAGE (pruner< X >)[inline]

Here is the call graph for this function:

4.37.2 Member Data Documentation

4.37.2.1 template<class X> X& Concepts::pruner< X >::i[private]

4.37.2.2 template<class X> refvector<ulong> Concepts::pruner< X >::v[private]

4.37.2.3 template<class X> const refvector<valerg> Concepts::pruner< X >::vc[private]

4.37.2.4 template<class X> const refvector<ulong> Concepts::pruner< X >::vd[private]

4.37.2.5 template<class X> refvector<valerg> Concepts::pruner< X >::ve[private]

4.37.2.6 template<class X> const X& Concepts::pruner< X >::Y[private]

The documentation for this class was generated from the following file:

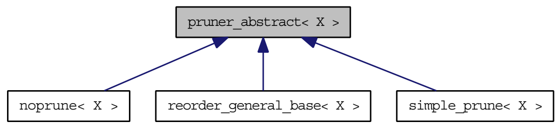
- concepts.hh

4.38 pruner_abstract< X > Class Template Reference

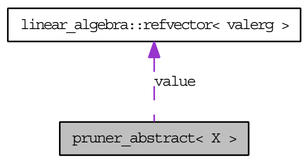
Abstract class implementing the Concepts::pruner concept.

#include <prunerabstract.h>

Inheritance diagram for pruner_abstract< X >:



Collaboration diagram for pruner_abstract< X >:



Public Member Functions

- pruner_abstract ()
  Default constructor.
- pruner_abstract (const pruner_abstract< X > &a)
  Copy constructor.
- valerg compute_property (ulong N) const
  Wrap the Library::compute_property to exclude pruned access.
- ulong get_space_size () const
  Adjust the space_size to reflect the absence of pruned values.
- ulong get_bits () const
  Compute the bits of the adjusted space.
- valerg get_value (ulong N) const
  Same as compute_property(ulong N), but absolute numbering.
- virtual ulong prune (double &lambda, ulong &conf1, ulong &conf2, long &config, const refvector< ulong > &visit) const =0

- ulong prune (double &lambda, ulong &conf1, ulong &conf2, long &config) const
- virtual ulong reprune (ulong N) const =0
  Given an "absolute" reference number return the current reference number.
- virtual ulong deprune (ulong N) const =0
  self-explanatory

Protected Attributes

- refvector< ulong > pruned_visited
  List of pruned indices in Library.
- refvector< ulong > & visited
  Reference to Library::visited.
- refvector< valerg > & value
  Reference to Library::value.
- bool bits_computed
- bool space_size_computed
- ulong space_size
- ulong bits

Private Member Functions

- BOOST_CONCEPT_ASSERT ((Concepts::Library< X >))

template<class X> class pruner_abstract< X >

4.38.1 Constructor & Destructor Documentation

4.38.1.1 template<class X> pruner_abstract< X >::pruner_abstract ()[inline]

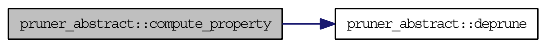
4.38.1.2 template<class X> pruner_abstract< X >::pruner_abstract (const pruner_abstract< X > & *a*)[nline]

4.38.2 Member Function Documentation

4.38.2.1 template<class X> pruner_abstract< X >::BOOST_CONCEPT_ASSERT ((Concepts::Library< X >))[private]

4.38.2.2 template<class X> valerg pruner_abstract< X >::compute_property (ulong *N*) const[inline]

Here is the call graph for this function:

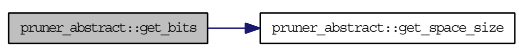


4.38.2.3 template<class X> virtual ulong pruner_abstract< X >::deprune (ulong *N*) const[pure virtual]

Implemented in noprune< X >, reorder_general_base< X >, and simple_prune< X >.

4.38.2.4 template<class X> ulong pruner_abstract< X >::get_bits () const[inline]

Here is the call graph for this function:

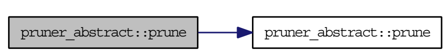


4.38.2.5 template<class X> ulong pruner_abstract< X >::get_space_size () const[inline]

4.38.2.6 template<class X> valerg pruner_abstract< X >::get_value (ulong *N*) const[inline]

4.38.2.7 template<class X> ulong pruner_abstract< X >::prune (double & *lambda*, ulong & *conf1*, ulong & *conf2*, long & *config*) const[inline]

Here is the call graph for this function:



4.38.2.8 template<class X> virtual ulong pruner_abstract< X >::prune (double & *lambda*, ulong & *conf1*, ulong & *conf2*, long & *config*, const refvector< ulong > & *visit*) const[pure virtual]

Implemented in noprune< X >, reorder_general_base< X >, and simple_prune< X >.

4.38.2.9 template<class X> virtual ulong pruner_abstract< X >::reprune (ulong *N*) const[pure virtual]

Implemented in noprune< X >, reorder_general_base< X >, and simple_prune< X >.

4.38.3 Member Data Documentation

4.38.3.1 template<class X> ulong pruner_abstract< X >::bits[mutable, protected]

4.38.3.2 template<class X> bool pruner_abstract< X >::bits_computed[mutable, protected]

4.38.3.3 template<class X> refvector<ulong> pruner_abstract< X >::pruned_visited[mutable, protected]

4.38.3.4 template<class X> ulong pruner_abstract< X >::space_size[mutable, protected]

4.38.3.5 template<class X> bool pruner_abstract< X >::space_size_computed[mutable, protected]

4.38.3.6 template<class X> refvector<valerg>& pruner_abstract< X >::value[mutable, protected]

4.38.3.7 template<class X> refvector<ulong>& pruner_abstract< X >::visited[mutable, protected]

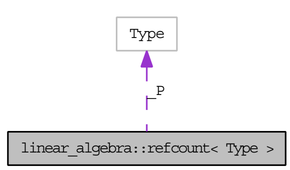
The documentation for this class was generated from the following file:

- prunerabstract.h

4.39 linear_algebra::refcount< Type > Class Template Reference

Class refcount provides reference counted pointers. Most basic use.

Collaboration diagram for linear_algebra::refcount< Type >:



Public Member Functions

- refcount ()
  Default constructor;.
- refcount (Type *l)
  Constructor from a pointer.
- refcount (const refcount< Type > &rhs)
  Copy constructor, but no deep copy.
- refcount (refcount< Type > &rhs)
  Copy constructor, but no deep copy.
- ~refcount ()
  Destructor checks for reference count and deletes if necessary.
- refcount< Type > & operator= (refcount< Type > &rhs)
  Assignment without deep copy. Old information is discarded if no references remain.
- operator Type * ()
  Let refcount behave like any other pointer.
- operator const Type * () const

Private Attributes

- unsigned long * _refcount
  Reference count.
- Type * _P
  Data pointer.

template<typename Type> class linear_algebra::refcount< Type >

4.39.1 Constructor & Destructor Documentation

4.39.1.1 template<class Type> linear_algebra::refcount< Type >::refcount ()

4.39.1.2 template<class Type> linear_algebra::refcount< Type >::refcount (Type * *l*)

4.39.1.3 template<class Type> linear_algebra::refcount< Type >::refcount (const refcount< Type > & *rhs*)

4.39.1.4 template<class Type> linear_algebra::refcount< Type >::refcount (refcount< Type > & *rhs*)

4.39.1.5 template<class Type> linear_algebra::refcount< Type >::~refcount ()

4.39.2 Member Function Documentation

4.39.2.1 template<class Type> linear_algebra::refcount< Type >::operator const Type * () const

4.39.2.2 template<class Type> linear_algebra::refcount< Type >::operator Type * ()

4.39.2.3 template<class Type> refcount< Type > & linear_algebra::refcount< Type >::operator= (refcount< Type > & *r*hs

4.39.3 Member Data Documentation

4.39.3.1 template<typename Type> Type* linear_algebra::refcount< Type >::_P[private]

4.39.3.2 template<typename Type> unsigned long* linear_algebra::refcount< Type >::_refcount[private]

The documentation for this class was generated from the following files:

- include/BCR_CPP_LA/refcount.decl
- include/BCR_CPP_LA/refcount.h

4.40 linear_algebra::refvector< TN > Class Template Reference

Class refvector provides reference counted vectors.

Public Member Functions

- ~refvector ()
  Destructor checks for reference count and deletes if necessary.
- void clear ()
- bool operator== (const refvector< TN > &a) const
  Compares two refvectors.

- template<class T> bool same (const refvector< T > &a) const

  Determines if two refvectors are exactly the same.

- refvector< TN > & operator= (const refvector< TN > &rhs)

  Assignment with necessary deep copy. Old information is discarded if no references remain.

- refvector< TN > & operator= (refvector< TN > &rhs)

  Assignment without deep copy. Old information is discarded if no references remain.

- refvector< TN > & operator= (const std::vector< TN > &rhs)

- refvector< TN > & set (long i, TN &rhs)

- void operator>> (std::ostream &o) const

- refvector< TN > & operator *= (const refvector< TN > &a)

  Multiply by a refvector.

- refvector< TN > element_prod (const refvector< TN > &a) const

  Multiply by a refvector.

- TN operator * (const refvector< TN > &b) const

  Scalar product of refvectors.

- template<class TN3, class TN2> TN3 & multiply (const refvector< TN2 > &b, TN3 &r) const

  Scalar product of refvectors.

- template<class TN3, class TN2> TN3 operator * (const refvector< TN2 > &b) const

  Scalar product of refvectors.

- refvector< TN > & operator+= (const refvector< TN > &B)

  Add a refvector.

- refvector< TN > operator+ (const refvector< TN > &B) const

  Add two refvectors.

- refvector< TN > & operator-= (const refvector< TN > &B)

  Subtract a refvector.

- refvector< TN > operator- (const refvector< TN > &B) const

  Subtract two refvectors.

- refvector< TN > & subtract (const refvector< TN > &B, refvector< TN > &r) const

  Subtract two refvectors.

- refvector< TN > & operator+= (const sparse_vector< TN > &a)
    Add a sparse vector.

- refvector< TN > & operator ∗= (const TN x)
    Multiply by an element.

- template<class TN2> refvector< TN > operator ∗= (const TN2 &a)
    Multiply by an element into the vector.

- template<class TN2> refvector< TN > operator ∗ (const TN2 x) const
    Multiply by an element.

- template<class TN2> refvector< TN > & multiply (const TN2 x, refvector< TN > &a) const
    Multiply by an element.

- template<class TN2> refvector< TN > operator/ (const TN2 x) const
    Divide by an element.

- void display (ostream &outs=cout) const
    Simple display of refvector.

- TN max () const
    Returns the maximum element.

- void normalise ()
    Normalise the vector in accordance with the $\| \cdot \|_2$ *2-norm.*

- TN & operator[ ] (long i)
    Direct access operator.

- const TN & operator[ ] (long i) const
    Direct access operator.

- std::vector< TN > ∗ operator → ()
    Dereferencing operator.

- const std::vector< TN > ∗ operator → () const
    Dereferencing operator.

- std::vector< TN > & vec ()
    Return the vector.

- const std::vector< TN > & vec () const
    Return the vector.

- refvector< TN > & copy (const refvector< TN > &i, const long j)
  Do a copy of depth j of i into a referenced counter. /item refvector< TN > & copy (const refvector< TN > &i)
  Copy i with full depth.

- long dim () const
  Returns the dimension of the vector.

- long size () const
  Returns the length of the vector.

- refvector< TN > & push_back (const TN &a)

- void insert (long n, const TN &a)

- void resize (long n, const TN &a)

- void resize (long n)

- void erase (typename std::vector< TN >::iterator a)

- void write_to (std::ostream &OUTFILE)
  Write a refvector to a stream.

- refvector< TN > & zero ()
  Set the refvector to the zero vector of current dimension.

- long contains (const TN &a) const
  Determines whether the refvector contains a certain element and returns its first occurence position.

- refvector< TN > & concat (const refvector< TN > &a)
  Appends another refvector to the end.

Constructors

- refvector ()
  Dereferencing operator.

- refvector (const refvector< TN > &rhs)
  Copy constructor necessarily deep to first level.

- refvector (refvector< TN > &rhs)
  Copy constructor, but no deep copy.

- refvector (long i)
  Initialise the vector to have i entries.
- refvector (const std::vector< TN > &i)
  Do a deep copy of i into a referenced counter.
- refvector (long j, const TN ∗rhs)
  Do a deep copy of i into a referenced counter.
- refvector (std::istream &IN)
  Read a refvector from a stream as writexttten by refvector::write_to().

Public Attributes

- char _deli

Private Member Functions

- const refvector< TN > & copy (const refvector< TN > &r) const
  Copy i with depth 1.
- const refvector< TN > & copy (const refvector< TN > &r, long i) const
  Do a copy of depth j of i into a referenced counter.
- const refvector< TN > & operator= (const refvector< TN > &r) const
  Assignment without deep copy. Old information is discarded if no references remain.

Private Attributes

- unsigned long ∗ _refcount
  Reference count.
- std::vector< TN > ∗ _
  Data pointer.

template<class TN> class linear_algebra::refvector< TN >

4.40.1 Constructor & Destructor Documentation

4.40.1.1 template<class TN> linear_algebra::refvector< TN >::refvector ()

Class refvector provides reference counted vectors. Default constructor;

4.40.1.2 template<class TN> linear_algebra::refvector< TN >::refvector (const refvector< TN > & *rhs*)

4.40.1.3 template<class TN> linear_algebra::refvector< TN >::refvector (refvector< TN > & *rhs*)

4.40.1.4 template<class TN> linear_algebra::refvector< TN >::refvector (long *i*)

4.40.1.5 template<class TN> linear_algebra::refvector< TN >::refvector (const std::vector< TN > & *i*)

4.40.1.6 template<class TN> linear_algebra::refvector< TN >::refvector (long *j*, const TN * *rhs*)

4.40.1.7 template<class TN> linear_algebra::refvector< TN >::refvector (std::istream & *IN*)

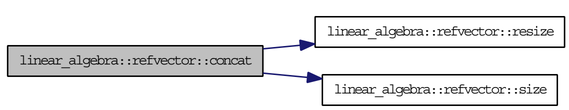
4.40.1.8 template<class TN> linear_algebra::refvector< TN >::~refvector ()

4.40.2 Member Function Documentation

4.40.2.1 template<class TN> void linear_algebra::refvector< TN >::clear ()

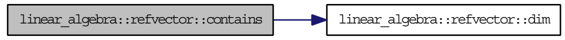
4.40.2.2 template<class TN> refvector< TN > & linear_algebra::refvector< TN >::concat (const refvector< TN > & *a*)

Here is the call graph for this function:



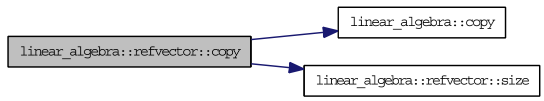
4.40.2.3 template<class TN> long linear_algebra::refvector< TN >::contains (const TN & *a*) const

Here is the call graph for this function:



4.40.2.4 template<class TN> const refvector< TN > & linear_algebra::refvector< TN >::copy (const refvector< TN > & r, long i) const[private]

Here is the call graph for this function:

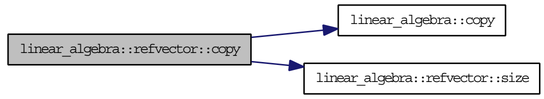


4.40.2.5 template<class TN> const refvector< TN > & linear_algebra::refvector< TN >::copy (const refvector< TN > & r) const[private]

Here is the call graph for this function:

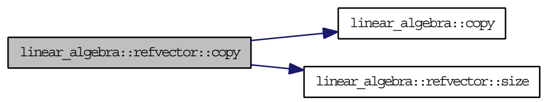
4.40.2.6 template<class TN> refvector< TN > & linear_algebra::refvector< TN >::copy (const refvector< TN > & i)

Here is the call graph for this function:



4.40.2.7 template<class TN> refvector< TN > & linear_algebra::refvector< TN >::copy (const refvector< TN > & i, const long j)

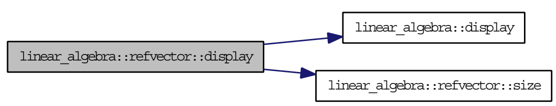
Here is the call graph for this function:



4.40.2.8 template<class TN> long linear_algebra::refvector< TN >::dim () const

4.40.2.9 template<class TN> void linear_algebra::refvector< TN >::display (ostream & *outs* = cout) const

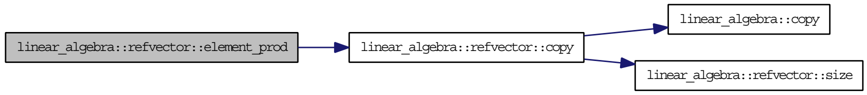
Here is the call graph for this function:



4.40.2.10 template<class TN> refvector< TN > linear_algebra::refvector< TN >::element_prod (const refvector< TN > & a) const

This is an elementwise multiplication which results in a new vector. $c_i = c_i \cdot a_i$. See refvector::operator*=().

Here is the call graph for this function:



4.40.2.11 template<class TN> void linear_algebra::refvector< TN >::erase (typename std::vector< TN >::iterator a)

4.40.2.12 template<class TN> void linear_algebra::refvector< TN >::insert (long n, const TN & a)

4.40.2.13 template<class TN> TN linear_algebra::refvector< TN >::max () const

Here is the call graph for this function:

4.40.2.14 template<class TN> template<class TN2> refvector< TN > & linear_algebra::refvector< TN >::multiply (const TN2 x, refvector< TN > & a) const

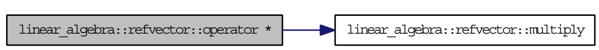
This may pose problems in some setextttings when TN∗TN2 is not defined.

4.40.2.15 template<class TN> template<class TN3, class TN2> TN3& linear_algebra::refvector< TN >::multiply (const refvector< TN2 > & b, TN3 & r) const

4.40.2.16 template<class TN> void linear_algebra::refvector< TN >::normalise ()

4.40.2.17 template<class TN> template<class TN2> refvector< TN > linear_algebra::refvector< TN >::operator ∗ (const TN2 x) const

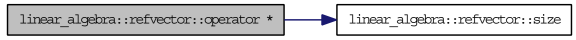
Here is the call graph for this function:



4.40.2.18 template<class TN> template<class TN3, class TN2> TN3 linear_algebra::refvector< TN >::operator ∗ (const refvector< TN2 > & b) const

4.40.2.19 template<class TN> TN linear_algebra::refvector< TN >::operator ∗ (const refvector< TN > & b) const

Here is the call graph for this function:



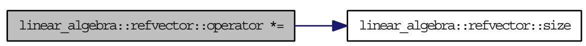
4.40.2.20 template<class TN> template<class TN2> refvector< TN > linear_algebra::refvector< TN >::operator *= (const TN2 & *a*)

4.40.2.21 template<class TN> refvector< TN > & linear_algebra::refvector< TN >::operator *= (const TN *x*)

4.40.2.22 template<class TN> refvector< TN > & linear_algebra::refvector< TN >::operator *= (const refvector< TN > & *a*)

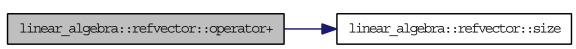
This is an elementwise multiplication which results in a new vector. $c_i = c_i \cdot a_i$. See also refvector::element_prod

Here is the call graph for this function:



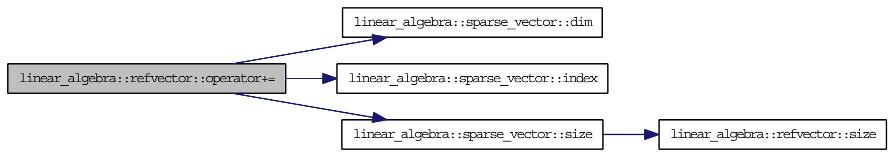
4.40.2.23 template<class TN> refvector< TN > linear_algebra::refvector< TN >::operator+ (const refvector< TN > & *B*) const

Here is the call graph for this function:



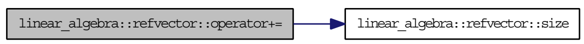
4.40.2.24 template<class TN> refvector< TN > & linear_algebra::refvector< TN >::operator+= (const sparse_vector< TN > & *a*)

Here is the call graph for this function:



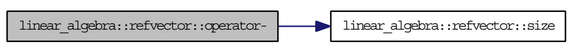
4.40.2.25 template<class TN> refvector< TN > & linear_algebra::refvector< TN >::operator+= (const refvector< TN > & *B*)

Here is the call graph for this function:



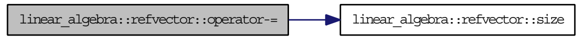
4.40.2.26 template<class TN> refvector< TN > linear_algebra::refvector< TN >::operator- (const refvector< TN > &) const

Here is the call graph for this function:



4.40.2.27 template<class TN> refvector< TN > & linear_algebra::refvector< TN >::operator-= (const refvector< TN > & *B*)

Here is the call graph for this function:



4.40.2.28 template<class TN> const vector< TN > * linear_algebra::refvector< TN >::operator → () const

4.40.2.29 template<class TN> vector< TN > * linear_algebra::refvector< TN >::operator → ()

4.40.2.30 template<class TN> template<class TN2> refvector< TN > linear_algebra::refvector< TN >::operator/ (const TN2 *x*) const

This may pose problems in some setextttings when TN*=TN2 is not defined.

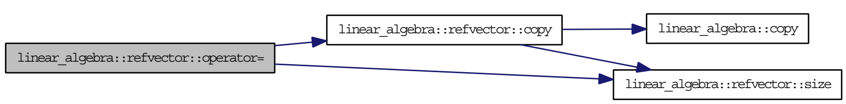
4.40.2.31 template<class TN> const refvector< TN > & linear_algebra::refvector< TN >::operator= (const refvector< TN > & *r*) const[private]

4.40.2.32 template<class TN> refvector<TN>& linear_algebra::refvector< TN >::operator= (const std::vector< TN > & *rhs*)

4.40.2.33 template<class TN> refvector< TN > & linear_algebra::refvector< TN >::operator= (refvector< TN > & *rhs*)

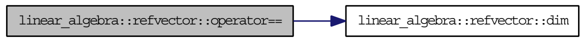
4.40.2.34 template<class TN> refvector< TN > & linear_algebra::refvector< TN >::operator= (const refvector< TN > & *rhs*)

Here is the call graph for this function:



4.40.2.35 template<class TN> bool linear_algebra::refvector< TN >::operator== (const refvector< TN > & *a*) const

Here is the call graph for this function:



4.40.2.36 template<class TN> void linear_algebra::refvector< TN >::operator>> (std::ostream & *o*) const

4.40.2.37 template<class TN> const TN & linear_algebra::refvector< TN >::operator[] (long *i*) cons

4.40.2.38 template<class TN> TN & linear_algebra::refvector< TN >::operator[] (long *i*)

4.40.2.39 template<class TN> refvector< TN > & linear_algebra::refvector< TN >::push_back (const TN & *a*)

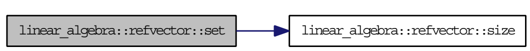
4.40.2.40 template<class TN> void linear_algebra::refvector< TN >::resize (long *n*)

4.40.2.41 template<class TN> void linear_algebra::refvector< TN >::resize (long *n*, const TN & *a*)

4.40.2.42 template<class TN> template<class T> bool linear_algebra::refvector< TN >::same (const refvector< T > & *a*) const

4.40.2.43 template<class TN> refvector< TN > & linear_algebra::refvector< TN >::set (long *i*, TN & *rhs*)

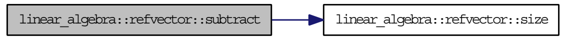
Here is the call graph for this function:



4.40.2.44 template<class TN> long linear_algebra::refvector< TN >::size () const

4.40.2.45 template<class TN> refvector< TN > & linear_algebra::refvector< TN >::subtract (const refvector< TN > & *B*, refvector< TN > & *r*) const

Here is the call graph for this function:



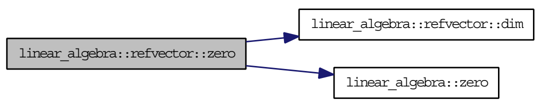
4.40.2.46 template<class TN> const vector< TN > & linear_algebra::refvector< TN >::vec () const

4.40.2.47 template<class TN> vector< TN > & linear_algebra::refvector< TN >::vec ()

4.40.2.48 template<class TN> void linear_algebra::refvector< TN >::write_to (std::ostream & *OUTFILE*)

4.40.2.49 template<class TN> refvector< TN > & linear_algebra::refvector< TN >::zero ()

Here is the call graph for this function:



4.40.3 Member Data Documentation

4.40.3.1 template<class TN> char linear_algebra::refvector< TN >::_delim

4.40.3.2 template<class TN> std::vector<TN>∗ linear_algebra::refvector< TN >::_P[private]

4.40.3.3 template<class TN> unsigned long∗ linear_algebra::refvector< TN >::_refcount[private]

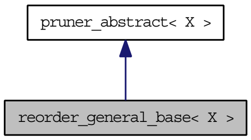
The documentation for this class was generated from the following files:

- include/BCR_CPP_LA/refcount.decl
- include/BCR_CPP_LA/refcount.h

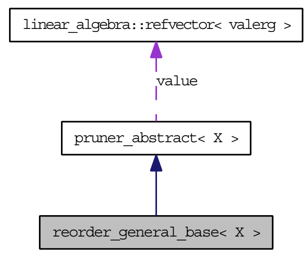
4.41 reorder_general_base< X > Class Template Reference

#include <reordergeneralbase.hh>

Inheritance diagram for reorder_general_base< X >:



Collaboration diagram for reorder_general_base< X >:



Public Member Functions

- reorder_general_base (reorder_general_base< X > &L)
  Copy Constructor.
- reorder_general_base (const reorder_general_base< X > &L)
  Copy Constructor.
- reorder_general_base (const refvector< long > &ibases)
  Constructor with a vector of bases.
- void adjust_lagrange (double &lambda, ulong &conf1, ulong &conf2, long &config, const refvector< ulong > &visited_run) const
  Compute the increase of lambda and assess current best value.
- ulong prune (double &lambda, ulong &conf1, ulong &conf2, long &config, const refvector< ulong > &visited_run) const
  Adjust the Lagrange multiplier and prune the library.
- ulong deprune (ulong N) const
  Reverse-map N to the global index.
- ulong reprune (ulong N) const
  Given an "absolute" reference number return the current reference number.

Private Attributes

- refvector< refvector< long > > base_order

  Records the order of digits in each base.

- refvector< refvector< valerg > > base_averages

  Records the average values associated with each digit and base.

- ulong oldvisitedsize

  Records the size of visited prior to the latest iteration.

4.41.1 Detailed Description

template<class X> class reorder_general_base< X >

This class takes an array of bases and orders the bases so as to produce a smooth optimization process. Each digit per base is weighted by the average computed values for compounds with the appropriate digit. There is no actual pruning done!

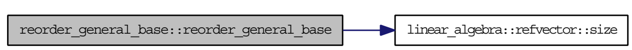
4.41.2 Constructor & Destructor Documentation

4.41.2.1 template<class X> reorder_general_base< X >::reorder_general_base (reorder_general_base< X > & *L*)[inline]

4.41.2.2 template<class X> reorder_general_base< X >::reorder_general_base (const reorder_general_base< X > & *L*)[inline]

4.41.2.3 template<class X> reorder_general_base< X >::reorder_general_base (const refvector< long > & *ibases*)[inline]

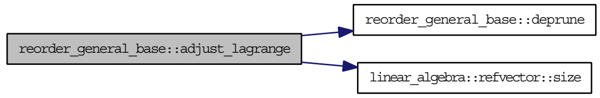
Here is the call graph for this function:



4.41.3 Member Function Documentation

4.41.3.1 template<class X> void reorder_general_base< X >::adjust_lagrange (double & *lambda*, ulong & *conf1*, ulong & *conf2*, long & *config*, const refvector< ulong > & *visited_run*) const[inline]

Here is the call graph for this function:



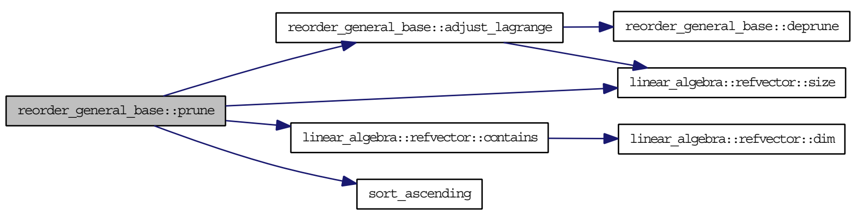
4.41.3.2 template<class X> ulong reorder_general_base< X >::deprune (ulong N) const[inline, virtual]

Implements pruner_abstract< X >.

4.41.3.3 template<class X> ulong reorder_general_base< X >::prune (double & *lambda*, ulon & *conf1*, ulong & *conf2*, long & *config*, const refvector< ulong > & *visited_run*) const[inline, virtual]

Implements pruner_abstract< X >.

Here is the call graph for this function:



4.41.3.4 template<class X> ulong reorder_general_base< X >::reprune (ulong *N*) const[inline, virtual]

Implements pruner_abstract< X >.

4.41.4 Member Data Documentation

4.41.4.1 template<class X> refvector< refvector<valerg> > reorder_general_base< X >::base_averages[mutable, private]

4.41.4.2 template<class X> refvector< refvector < long > > reorder_general_base< X >::base_order[mutable, private]

4.41.4.3 template<class X> ulong reorder_general_base< X >::oldvisitedsize[mutable, private]

The documentation for this class was generated from the following file:

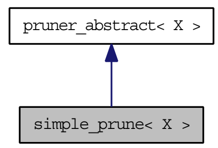
- reordergeneralbase.hh

4.42 simple_prune< X > Class Template Reference

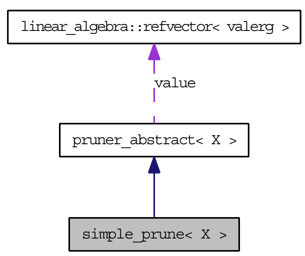
Class to prune a Library.

#include <simpleprune.h>

Inheritance diagram for simple_prune< X >:



Collaboration diagram for simple_prune< X >:



Public Member Functions

- simple_prune ()
- simple_prune (const simple_prune< X > &a)
- ulong prune (double &lambda, ulong &conf1, ulong &conf2, long &config, const refvector< ulong > &a) const
  Prune the Library and adjust lambda. Pruned entries are in pruned_visited.
- ulong deprune (ulong N) const
  Revert an index from the pruned Library to the original Library.
- ulong reprune (ulong N) const
  Convert an "absolute" index to a relative index.

template<class X> class simple_prune< X >

4.42.1 Constructor & Destructor Documentation

4.42.1.1 template<class X> simple_prune< X >::simple_prune ()[inline]

4.42.1.2 template<class X> simple_prune< X >::simple_prune (const simple_prune< X > & *a*)[inline]

4.42.2 Member Function Documentation

4.42.2.1 template<class X> ulong simple_prune< X >::deprune (ulong *N*) const[virtual]

Implements pruner_abstract< X >.

4.42.2.2 template<class X> ulong simple_prune< X >::prune (double & *lambda*, ulong & *conf1*, ulong & *conf2*, long & *config*, const refvector< ulong > & *visited_run*) const[virtual]

Inferior values have larger penalties than the current best. Lambda is adjusted such that $Prop_{conf1} - \lambda' Pen_{conf1} < Prop_j - \lambda' Pen_j$ for some j and $\lambda' > \lambda$

Implements pruner_abstract< X >.

4.42.2.3 template<class X> ulong simple_prune< X >::reprune (ulong *N*) const[virtual]

Implements pruner_abstract< X >.

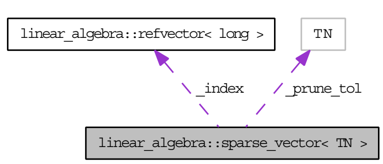
The documentation for this class was generated from the following files:

- simpleprune.h
- simpleprune.cc

4.43 linear_algebra::sparse_vector< TN > Class Template Reference

A class for sparse vectors.

Collaboration diagram for linear_algebra::sparse_vector< TN >:



Public Member Functions

- sparse_vector ()
  Default constructor. Dimension 0.
- sparse_vector (const long n)
  Constructs a sparse vector of dimension n without initialisation.
- sparse_vector (const long n, const std::vector< long > &a, const std::vector< TN > &b)
  Constructs a sparse vector of dimension n with initialisation.
- sparse_vector (const long n, const refvector< long > &a, const std::vector< TN > &b)
  Constructs a sparse vector of dimension n with initialisation.
- sparse_vector (const long n, const refvector< long > &a, const refvector< TN > &b)
  Constructs a sparse vector of dimension n with initialisation.
- sparse_vector (const long n, const std::vector< long > &a, const refvector< TN > &b)
  Constructs a sparse vector of dimension n with initialisation.
- sparse_vector (const sparse_vector &a)
  Copy constructor.—
- *sparse_vector*< TN > & operator *= (const sparse_vector< TN > &a)
  Multiply by a sparse vector.
- TN operator * (const sparse_vector &a) const
  Scalar product of sparse vectors.
- bool is_nonzero (const long x, long &i) const
  is (x) non zero
- TN operator[ ] (const long x)
  Direct access operator.
- TN operator[ ] (const long x) const
  Direct access operator.
- const TN & operator() (const long x) const
  Access by element operator.
- TN operator() (const long x)
  Access by element operator.

- sparse_vector< TN > & add (const long x, TN z)
  Add an element for consistency.
- sparse_vector< TN > & multiply (const long x, TN z)
  Multiply an element for consistency.
- sparse_vector< TN > & operator+= (const sparse_vector< TN > &B)
  Add a sparse vector.
- sparse_vector< TN > & operator-= (const sparse_vector< TN > &B)
  Subtract a sparse vector.
- sparse_vector< TN > operator- () const
  Subtract a sparse vector.
- sparse_vector< TN > & operator *= (const TN x)
  Multiply by an element.
- sparse_vector< TN > operator * (const TN x) const
  Multiply by an element.
- sparse_vector< TN > & operator/= (const TN x)
  Divide by an element.
- sparse_vector< TN > operator/ (const TN x) const
  Divide by an element.
- sparse_vector< TN > & operator= (const sparse_vector< TN > &b)
  Copy const sparse_vector.
- sparse_vector< TN > & operator= (sparse_vector< TN > &b)
  Copy const sparse_vector.
- sparse_vector< TN > & operator= (const std::vector< TN > b)
  Copy a const vector into a sparse vector.
- sparse_vector< TN > copy (const sparse_vector< TN > &b, long i)
  Copy sparse vector with copy of depth i.
- sparse_vector< TN > copy (const sparse_vector< TN > &b)
  Copy a sparse vector with depth=1.
- void set (long i, TN value)
- void prune (const TN x)
  Prune a sparse vector of negligible values.

- void prune ()
- bool display () const
  Simple display of sparse vector.
- TN max () const
  Returns the maximum element.
- void normalise ()
  Normalise the sparse vector in accordance with the $\|\cdot\|_2$ *2-norm.*
- long size () const
  Returns the number of nonzero elements.
- long index (long i) const
  Returns the index of the ith nonzero element.
- long dim () const
  Returns the dimension of the vector.
- void clear ()
  Releases the associated memory, i.e., sets the vector to 0.
- void zero ()
  Retains the dimension but releases the associated memory, i.e., sets the vector to 0.

Public Attributes

- TN _prune_tol
  Pruning tolerance.

Private Attributes

- long _size
  Dimension.
- refvector< TN > _vals
  Vector of values.
- linear_algebra::refvector< long > _index
  Vector of indices.

Friends

- class mat_sparse< TN >
- class mat_sym_sparse< TN >
- class mat_asym_sparse< TN >
- class mat_sym_full< TN >
- class mat_full< TN >

template<class TN> class linear_algebra::sparse_vector< TN >

4.43.1 Constructor & Destructor Documentation

4.43.1.1 template<class TN> linear_algebra::sparse_vector< TN >::sparse_vector ()

4.43.1.2 template<class TN> linear_algebra::sparse_vector< TN >::sparse_vector (const long *n*)

4.43.1.3 template<class TN> linear_algebra::sparse_vector< TN >::sparse_vector (const long *n*, const std::vector< long > & *a*, const std::vector< TN > & *b*)

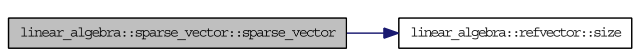
4.43.1.4 template<class TN> linear_algebra::sparse_vector< TN >::sparse_vector (const long *n*, const refvector< long > & *a*, const std::vector< TN > & *b*)

4.43.1.5 template<class TN> linear_algebra::sparse_vector< TN >::sparse_vector (const long *n*, const refvector< long > & *a*, const refvector< TN > & *b*)

Parameters:

a Reference counted vector indices.
Reference counted ector values.

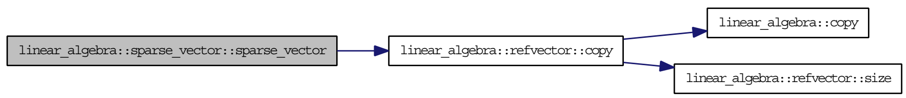
Here is the call graph for this function:



4.43.1.6 template<class TN> linear_algebra::sparse_vector< TN >::sparse_vector (const long *n*, const std::vector< long > & *a*, const refvector< TN > & *b*)

4.43.1.7 template<class TN> linear_algebra::sparse_vector< TN >::sparse_vector (const sparse_vector< TN > & a)

Here is the call graph for this function:

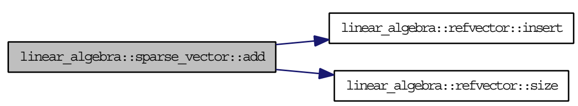


4.43.2 Member Function Documentation

4.43.2.1 template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::add (const long x, TN z)

One of the pitfalls of using this for assignments is that if _size is set incorrectly the throw argument will result in an SIGABORT if not caught.

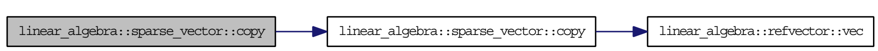
Here is the call graph for this function:



4.43.2.2 template<class TN> void linear_algebra::sparse_vector< TN >::clear ()

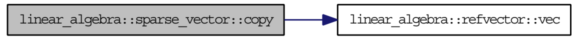
4.43.2.3 template<class TN> sparse_vector< TN > linear_algebra::sparse_vector< TN >::copy (const sparse_vector< TN > & b)

Here is the call graph for this function:



4.43.2.4 template<class TN> sparse_vector< TN > linear_algebra::sparse_vector< TN >::copy (const sparse_vector< TN > & b, long i)

Here is the call graph for this function:



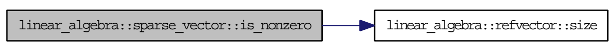
4.43.2.5 template<class TN> long linear_algebra::sparse_vector< TN >::dim () const

4.43.2.6 template<class TN> bool linear_algebra::sparse_vector< TN >::display () const

4.43.2.7 template<class TN> long linear_algebra::sparse_vector< TN >::index (long i) const

4.43.2.8 template<class TN> bool linear_algebra::sparse_vector< TN >::is_nonzero (const long x, long & i) const

Here is the call graph for this function:



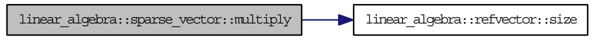
4.43.2.9 template<class TN> TN linear_algebra::sparse_vector< TN >::max () const

Here is the call graph for this function:

4.43.2.10 template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::multiply (const long x, TN z)

One of the pitfalls of using this for assignments is that if _size is set incorrectly the throw argument will result in an SIGABORT if not caught.

Here is the call graph for this function:



4.43.2.11 template<class TN> void linear_algebra::sparse_vector< TN >::normalise ()

4.43.2.12 template<class TN> sparse_vector< TN > linear_algebra::sparse_vector< TN >::operator $*$ (const TN x) const

Here is the call graph for this function:

4.43.2.13 template<class TN> TN linear_algebra::sparse_vector< TN >::operator $*$ (const sparse_vector< TN > & a) const

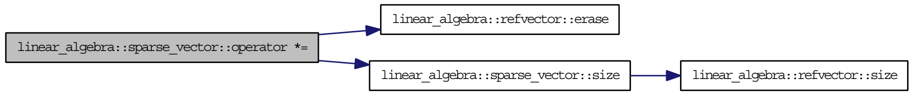
Here is the call graph for this function:

4.43.2.14 template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::operator $*=$ (const TN x)

4.43.2.15 template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::operator $*=$ (const sparse_vector< TN > & a)

This is an elementwise multiplication which results in a new vector. $c_i = c_i \cdot a_i$.

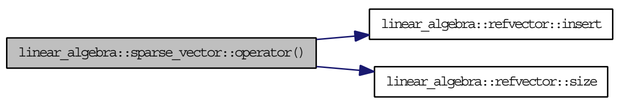
Here is the call graph for this function:



4.43.2.16 template<class TN> TN & linear_algebra::sparse_vector< TN >::operator() (const long *x*)

One of the pitfalls of using this for assignments is that if _size is set incorrectly the throw argument will result in an SIGABORT if not caught.

Here is the call graph for this function:

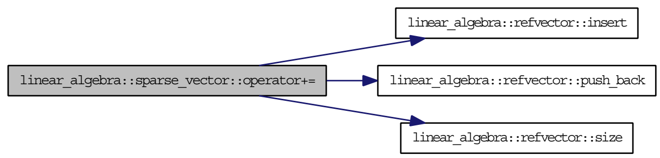


4.43.2.17 template<class TN> const TN & linear_algebra::sparse_vector< TN >::operator() (const long *x*) const

Here is the call graph for this function:

4.43.2.18 template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::operator+= (const sparse_vector< TN > & *B*)

Here is the call graph for this function:

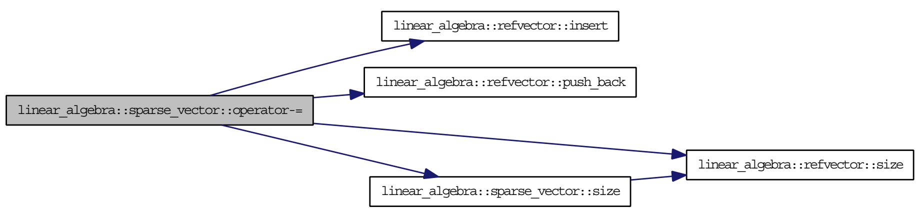


4.43.2.19 template<class TN> sparse_vector< TN > linear_algebra::sparse_vector< TN >::operator- () const

Test

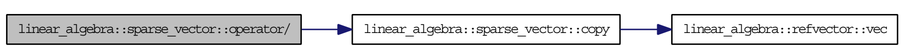
4.43.2.20 template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::operator-= (const sparse_vector< TN > & B)

Here is the call graph for this function:



4.43.2.21 template<class TN> sparse_vector< TN > linear_algebra::sparse_vector< TN >::operator/ (const TN x) const

Here is the call graph for this function:



4.443.2.22 template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::operator/= (const TN x)

4.43.2.23 template<class TN> sparse_vector<TN>& linear_algebra::sparse_vector< TN >::operator= (const std::vector< TN > b)

4.43.2.24 template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::operator= (sparse_vector< TN > & b)

4.43.2.25 template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::operator= (const sparse_vector< TN > & b)

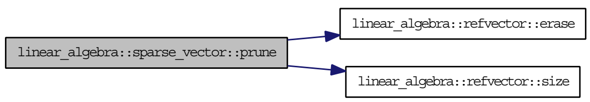
4.43.2.26 template<class TN> TN linear_algebra::sparse_vector< TN >::operator[] (const long *x*) const

4.43.2.27 template<class TN> TN & linear_algebra::sparse_vector< TN >::operator[] (const long *x*)

4.43.2.28 template<class TN> void linear_algebra::sparse_vector< TN >::prune ()

4.43.2.29 template<class TN> void linear_algebra::sparse_vector< TN >::prune (const TN *x*)

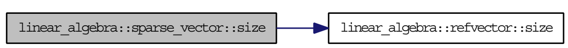
Here is the call graph for this function:



4.43.2.30 template<class TN> void linear_algebra::sparse_vector< TN >::set (long *i*, TN *value*)

4.43.2.31 template<class TN> long linear_algebra::sparse_vector< TN >::size () const

Here is the call graph for this function:



4.43.2.32 template<class TN> void linear_algebra::sparse_vector< TN >::zero ()

4.43.3 Friends and Related Function Documentation

4.43.3.1 template<class TN> friend class mat_asym_sparse< TN >[friend]

4.43.3.2 template<class TN> friend class mat_full< TN >[friend]

4.43.3.3 template<class TN> friend class mat_sparse< TN >[friend]

4.43.3.4 template<class TN> friend class mat_sym_full< TN >[friend]

4.43.3.5 template<class TN> friend class mat_sym_sparse< TN >[friend]

4.43.4 Member Data Documentation

4.43.4.1 template<class TN> linear_algebra::refvector< long > linear_algebra::sparse_vector< TN >::_index[private]

4.43.4.2 template<class TN> TN linear_algebra::sparse_vector< TN >::_prune_tol

4.43.4.3 template<class TN> long linear_algebra::sparse_vector< TN >::_size[private]

4.43.4.4 template<class TN> refvector<TN> linear_algebra::sparse_vector< TN >::_vals[private]

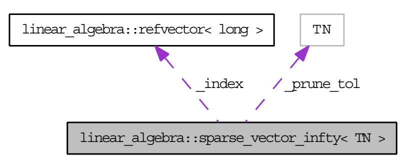
The documentation for this class was generated from the following files:

- include/BCR_CPP_LA/sparse_vector.decl
- include/BCR_CPP_LA/sparse_vector.h

4.44 linear_algebra::sparse_vector_infty< TN > Class Template Reference

A class for sparse vectors.

Collaboration diagram for linear_algebra::sparse_vector_infty< TN >:



Public Member Functions

- sparse_vector_infty ()
  Default constructor. Dimension 0.
- sparse_vector_infty (const std::vector< long > &a, const std::vector< TN > &b)
  Constructs an infinite sparse vector with initialisation.
- sparse_vector_infty (const refvector< long > &a, const std::vector< TN > &b)
  Constructs an infinite sparse vector with initialisation.
- sparse_vector_infty (const refvector< long > &a, const refvector< TN > &b)
  Constructs a sparse vector of infinite dimension with initialisation.

- sparse_vector_infty (const std::vector< long > &a, const refvector< TN > &b)
  Constructs an infinite sparse vector with initialisation.
- sparse_vector_infty (const sparse_vector_infty &a)
  Copy constructor.
- sparse_vector_infty< TN > & operator *= (const sparse_vector_infty< TN > &a)
  Multiply by a sparse vector.
- TN operator * (const sparse_vector_infty &a) const
  Scalar product of sparse vectors.
- bool is_nonzero (const long x, long &i) const
  is (x) non zero
- TN & operator[ ] (const long x)
  Direct access operator.
- TN operator[ ] (const long x) const
  Direct access operator.
- TN operator() (const long x) const
  Access by element operator.
- TN & operator() (const long x)
  Access by element operator.
- sparse_vector_infty< TN > & add (const long x, TN z)
  Add an element for consistency.
- sparse_vector_infty< TN > & multiply (const long x, TN z)
  Multiply an element for consistency.
- sparse_vector_infty< TN > & operator+= (const sparse_vector_infty< TN > &B)
  Add an infinite sparse vector.
- sparse_vector_infty< TN > & operator-= (const sparse_vector_infty< TN > &B)
  Subtract a sparse vector.
- sparse_vector_infty< TN > operator- () const
  Subtract a sparse vector.
- sparse_vector_infty< TN > & operator *= (const TN x)
  Multiply by an element.
- sparse_vector_infty< TN > operator * (const TN x) const
  Multiply by an element.

- sparse_vector_infty< TN > & operator/= (const TN x)
  Divide by an element.
- sparse_vector_infty< TN > operator/ (const TN x) const
  Divide by an element.
- sparse_vector_infty< TN > & operator= (const sparse_vector_infty< TN > &b)
  Copy const sparse_vector_infty.
- sparse_vector_infty< TN > & operator= (sparse_vector_infty< TN > &b)
  Copy const sparse_vector_infty.
- sparse_vector_infty< TN > & operator= (const std::vector< TN > b)
  Copy a const vector into a sparse vector.
- sparse_vector_infty< TN > copy (const sparse_vector_infty< TN > &b, long i)
  Copy sparse vector with copy of depth i.
- sparse_vector_infty< TN > copy (const sparse_vector_infty< TN > &b)
  Copy a sparse vector with depth=1.
- void set (long i, const TN &value)
- void set (long i, TN &value)
- void prune (const TN x)
  Prune a sparse vector of negligible values.
- void prune ()
- bool display () const
  Simple display of sparse vector.
- TN max () const
  Returns the maximum element.
- void normalise ()
  Normalise the sparse vector in accordance with the $\|\cdot\|_2$ *2-norm.*
- long size () const
  Returns the number of nonzero elements.
- long index (long i) const
  Returns the index of the ith nonzero element.
- void clear ()
  Releases the associated memory, i.e., sets the vector to 0.

- void zero ()
  Retains the dimension but releases the associated memory, i.e., sets the vector to 0.

Public Attributes

- TN _prune_tol
  Pruning tolerance.

Private Attributes

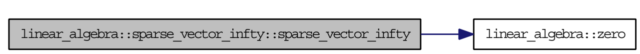
- long _size
  Non-zero entries.
- refvector< TN > _vals
  Vector of values.
- refvector< long > _index
  Vector of indices.

template<class TN> class linear_algebra::sparse_vector_infty< TN >

4.44.1 Constructor & Destructor Documentation

4.44.1.1 template<class TN> linear_algebra::sparse_vector_infty< TN >::sparse_vector_infty ()

Here is the call graph for this function:



4.44.1.2 template<class TN> linear_algebra::sparse_vector_infty< TN >::sparse_vector_infty (const std::vector< long > & *a*, const std::vector< TN > & *b*)

4.44.1.3 template<class TN> linear_algebra::sparse_vector_infty< TN >::sparse_vector_infty (const refvector< long > & *a*, const std::vector< TN > & *b*)

4.44.1.4 template<class TN> linear_algebra::sparse_vector_infty< TN >::sparse_vector_infty (const refvector< long > & *a*, const refvector< TN > & *b*)

Parameters:

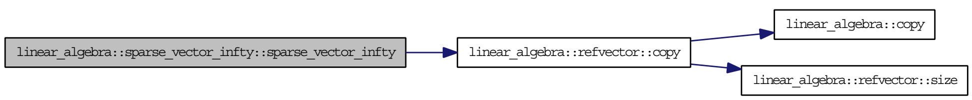
> *a* Reference counted vector indices.
> *b* Reference counted ector values.

Here is the call graph for this function:

4.44.1.5 template<class TN> linear_algebra::sparse_vector_infty< TN >::sparse_vector_infty (const std::vector< long > & *a*, const refvector< TN > & *b*)

4.44.1.6 template<class TN> linear_algebra::sparse_vector_infty< TN >::sparse_vector_infty (const sparse_vector_infty< TN > & *a*)

Here is the call graph for this function:

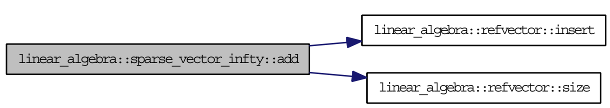


4.44.2 Member Function Documentation

4.44.2.1 template<class TN> sparse_vector_infty< TN > & linear_algebra::sparse_vector_infty< TN >::add (const long *x*, TN *z*)

One of the pitfalls of using this for assignments is that if _size is set incorrectly the throw argument will result in an SIGABORT if not caught.

Here is the call graph for this function:



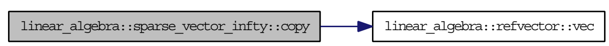
4.44.2.2 template<class TN> void linear_algebra::sparse_vector_infty< TN >::clear ()

4.44.2.3 template<class TN> sparse_vector_infty< TN > linear_algebra::sparse_vector_infty< TN >::copy (const sparse_vector_infty< TN > & b)

Here is the call graph for this function:

4.44.2.4 template<class TN> sparse_vector_infty< TN > linear_algebra::sparse_vector_infty< TN >::copy (const sparse_vector_infty< TN > & b, long i)

Here is the call graph for this function:

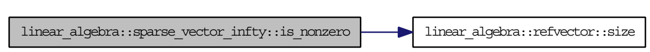


4.44.2.5 template<class TN> bool linear_algebra::sparse_vector_infty< TN >::display () const

4.44.2.6 template<class TN> long linear_algebra::sparse_vector_infty< TN >::index (long i) const

4.44.2.7 template<class TN> bool linear_algebra::sparse_vector_infty< TN >::is_nonzero (const long x, long & i) const

Here is the call graph for this function:



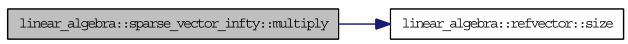
4.44.2.8 template<class TN> TN linear_algebra::sparse_vector_infty< TN >::max () const

Here is the call graph for this function:

4.44.2.9 template<class TN> sparse_vector_infty< TN > & linear_algebra::sparse_vector_infty< TN >::multiply (const long x, TN z)

One of the pitfalls of using this for assignments is that if _size is set incorrectly the throw argument will result in an SIGABORT if not caught.

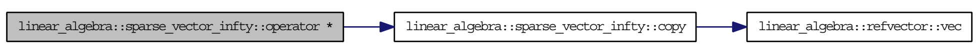
Here is the call graph for this function:



4.44.2.10 template<class TN> void linear_algebra::sparse_vector_infty< TN >::normalise ()

4.44.2.11 template<class TN> sparse_vector_infty< TN > linear_algebra::sparse_vector_infty< TN >::operator * (const TN x) const

Here is the call graph for this function:



4.44.2.12 template<class TN> TN linear_algebra::sparse_vector_infty< TN >::operator * (const sparse_vector_infty< TN > & a) const

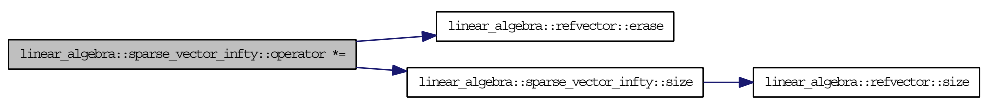
Here is the call graph for this function:

4.44.2.13 template<class TN> sparse_vector_infty< TN > & linear_algebra::sparse_vector_infty< TN >::operator *= (const TN x)

4.44.2.14 template<class TN> sparse_vector_infty< TN > & linear_algebra::sparse_vector_infty< TN >::operator *= (const sparse_vector_infty< TN > & a)

This is an elementwise multiplication which results in a new vector. $c_i = c_i \cdot a_i$.

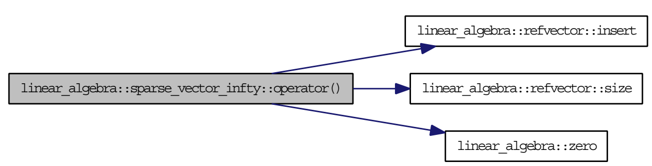
Here is the call graph for this function:



4.44.2.15 template<class TN> TN & linear_algebra::sparse_vector_infty< TN >::operator() (const long x)

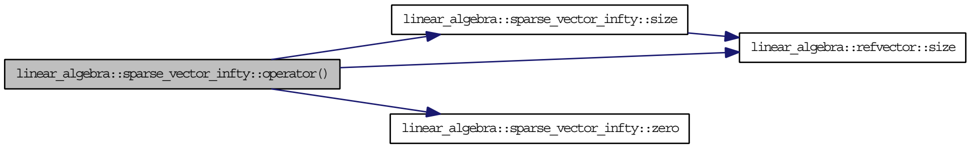
One of the pitfalls of using this for assignments is that if _size is set incorrectly the throw argument will result in an SIGABORT if not caught.

Here is the call graph for this function:



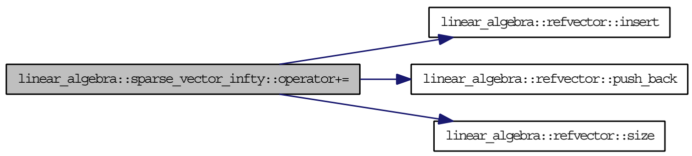
4.44.2.16 template<class TN> TN linear_algebra::sparse_vector_infty< TN >::operator() (const long x) const

Here is the call graph for this function:



4.44.2.17 template<class TN> sparse_vector_infty< TN > & linear_algebra::sparse_vector_infty< TN >::operator+= (const sparse_vector_infty< TN > & B)

Here is the call graph for this function:

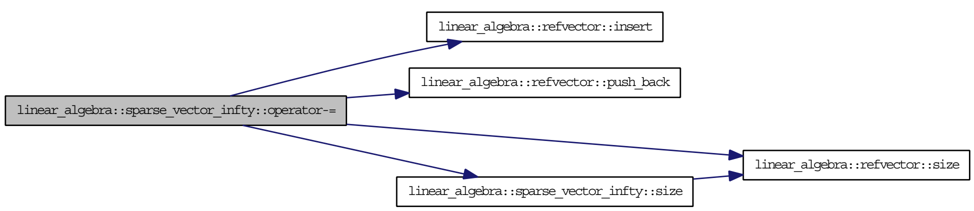


4.44.2.18 template<class TN> sparse_vector_infty< TN > linear_algebra::sparse_vector_infty< TN >::operator- () const

Test

4.44.2.19 template<class TN> sparse_vector_infty< TN > & linear_algebra::sparse_vector_infty< TN >::operator-= (const sparse_vector_infty< TN > & B)

Here is the call graph for this function:



4.44.2.20 template<class TN> sparse_vector_infty< TN > linear_algebra::sparse_vector_infty< TN >::operator/ (const TN x) const

Here is the call graph for this function:

4.44.2.21 template<class TN> sparse_vector_infty< TN > & linear_algebra::sparse_vector_infty< TN >::operator/= (const TN *x*)

4.44.2.22 template<class TN> sparse_vector_infty<TN>& linear_algebra::sparse_vector_infty< TN >::operator= (const std::vector< TN > *b*)

4.44.2.23 template<class TN> sparse_vector_infty< TN > & linear_algebra::sparse_vector_infty< TN >::operator= (sparse_vector_infty< TN > & *b*)

4.44.2.24 template<class TN> sparse_vector_infty< TN > & linear_algebra::sparse_vector_infty< TN >::operator= (const sparse_vector_infty< TN > & *b*)

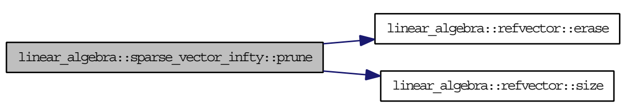
4.44.2.25 template<class TN> TN linear_algebra::sparse_vector_infty< TN >::operator[] (const long *x*) const

4.44.2.26 template<class TN> TN & linear_algebra::sparse_vector_infty< TN >::operator[] (const long *x*)

4.44.2.27 template<class TN> void linear_algebra::sparse_vector_infty< TN >::prune ()

4.44.2.28 template<class TN> void linear_algebra::sparse_vector_infty< TN >::prune (const TN *x*)

Here is the call graph for this function:

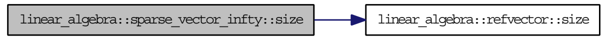


4.44.2.29 template<class TN> void linear_algebra::sparse_vector_infty< TN >::set (long *i*, TN & *value*)

4.44.2.30 template<class TN> void linear_algebra::sparse_vector_infty< TN >::set (long *i*, const TN & *value*)

4.44.2.31 template<class TN> long linear_algebra::sparse_vector_infty< TN >::size () const

Here is the call graph for this function:



4.44.2.32 template<class TN> void linear_algebra::sparse_vector_infty< TN >::zero ()

4.44.3 Member Data Documentation

4.44.3.1 template<class TN> refvector<long> linear_algebra::sparse_vector_infty< TN >::_index[private]

4.44.3.2 template<class TN> TN linear_algebra::sparse_vector_infty< TN >::_prune_tol

4.44.3.3 template<class TN> long linear_algebra::sparse_vector_infty< TN >::_sizetext[private]

4.44.3.4 template<class TN> refvector<TN> linear_algebra::sparse_vector_infty< TN >::_vals[private]

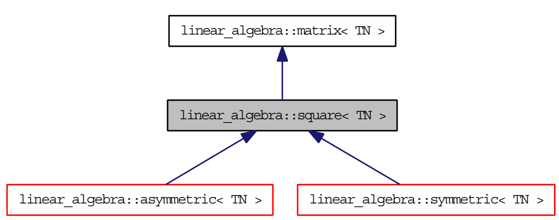
The documentation for this class was generated from the following files:

- include/BCR_CPP_LA/sparse_vector_infty.decl
- include/BCR_CPP_LA/sparse_vector_infty.h

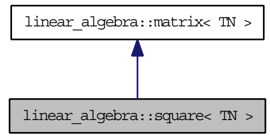
4.45 linear_algebra::square< TN > Class Template Reference

The square matrix class contains all n by n matrices.

Inheritance diagram for linear_algebra::square< TN >:



Collaboration diagram for linear_algebra::square< TN >:



Public Member Functions

- TN trace () const
  The square matrix class contains all n by n matrices.

template<class TN> class linear_algebra::square< TN >

4.45.1 Member Function Documentation

4.45.1.1 template<class TN> TN linear_algebra::square< TN >::trace () const

Reimplemented from linear_algebra::matrix< TN >.

Reimplemented in linear_algebra::mat_sym_sparse< TN >.

The documentation for this class was generated from the following files:

- include/BCR_CPP_LA/square.decl
- include/BCR_CPP_LA/square.h

4.46 boost::stateful_thread_group Class Reference

#include <stateful_thread_group.hh>

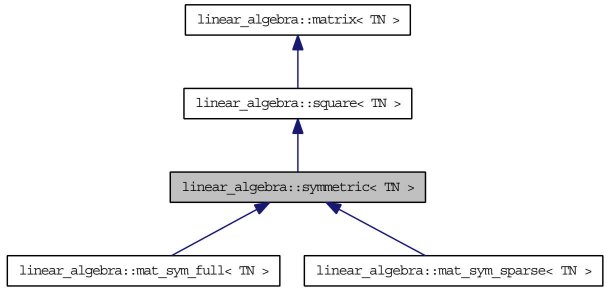
The documentation for this class was generated from the following file:

- include/stateful_thread_group.hh

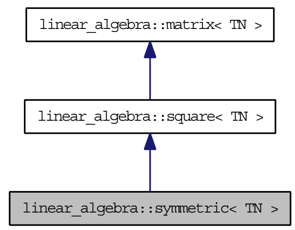
4.47 linear_algebra::symmetric< TN > Class Template Reference

The symmetric class contains all symmetric matrices.

Inheritance diagram for linear_algebra::symmetric< TN >:



Collaboration diagram for linear_algebra::symmetric< TN >:



- matrix< TN > & transpose (matrix< TN > &r) const

  The symmetric class contains all symmetric matrices.

- matrix< TN > & transpose (matrix< TN > &r)
- symmetric< TN > & transpose (symmetric< TN > &r) const
- symmetric< TN > & transpose (symmetric< TN > &r)
- symmetric< TN > & transpose () const
- symmetric< TN > & transpose ()
- matrix< TN > & copy (const matrix< TN > &a)
- matrix< TN > & copy (const matrix< TN > &a, const long i)
- matrix< TN > & copy (matrix< TN > &a, const long i)
- virtual symmetric< TN > & copy (const symmetric< TN > &a)=0
- virtual symmetric< TN > & copy (const symmetric< TN > &a, const long i)=0
- virtual symmetric< TN > & copy (symmetric< TN > &a, const long i)=0

- bool add (long x, long y, const TN z)
  Adds a value.

- symmetric< TN > & prune (const TN x=0) *Deletes all entries that are smaller in magnitude than x.*

template<class TN> class linear_algebra::symmetric< TN >

4.47.1 Member Function Documentation

4.47.1.1 template<class TN> bool linear_algebra::symmetric< TN >::add (long x, long y, const TN z)

Reimplemented in linear_algebra::mat_sym_full< TN >, linear_algebra::mat_sym_sparse< TN >, and linear_algebra::mat_sym_full< valerg >.

4.47.1.2 template<class TN> virtual symmetric<TN>& linear_algebra::symmetric< TN >::copy (symmetric< TN > & a, const long i)[pure virtual]

Implemented in linear_algebra::mat_sym_full< TN >, linear_algebra::mat_sym_sparse< TN >, and linear_algebra::mat_sym_full< valerg >.

4.47.1.3 template<class TN> virtual symmetric<TN>& linear_algebra::symmetric< TN >::copy (const symmetric< TN > & a, const long i)[pure virtual]

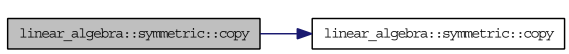
Implemented in linear_algebra::mat_sym_full< TN >, linear_algebra::mat_sym_sparse< TN >, and linear_algebra::mat_sym_full< valerg >.

4.47.1.4 template<class TN> virtual symmetric<TN>& linear_algebra::symmetric< TN >::copy (const symmetric< TN > & a)[pure virtual]

Implemented in linear_algebra::mat_sym_full< TN >, linear_algebra::mat_sym_sparse< TN >, and linear_algebra::mat_sym_full< valerg >.

4.47.1.5 template<class TN> matrix< TN > & linear_algebra::symmetric< TN >::copy (matrix< TN > & a, const long i)

Here is the call graph for this function:



4.47.1.6 template<class TN> matrix< TN > & linear_algebra::symmetric< TN >::copy (const matrix< TN > & a, const long i)

Here is the call graph for this function:

4.47.1.7 template<class TN> matrix< TN > & linear_algebra::symmetric< TN >::copy (const matrix< TN > & a)

4.47.1.8 template<class TN> symmetric<TN>& linear_algebra::symmetric< TN >::prune (const TN $x = 0$)

Reimplemented in linear_algebra::mat_sym_full< TN >, linear_algebra::mat_sym_sparse< TN >, and linear_algebra::mat_sym_full< valerg >.

4.47.1.9 template<class TN> symmetric< TN > & linear_algebra::symmetric< TN >::transpose ()

4.47.1.10 template<class TN> symmetric< TN > & linear_algebra::symmetric< TN >::transpose () const

4.47.1.11 template<class TN> symmetric< TN > & linear_algebra::symmetric< TN >::transpose (symmetric< TN > & r)

4.47.1.12 template<class TN> symmetric< TN > & linear_algebra::symmetric< TN >::transpose (symmetric< TN > & r) const

4.47.1.13 template<class TN> matrix< TN > & linear_algebra::symmetric< TN >::transpose (matrix< TN > & r)

4.47.1.14 template<class TN> matrix< TN > & linear_algebra::symmetric< TN >::transpose (matrix< TN > & r) const

The documentation for this class was generated from the following files:

- include/BCR_CPP_LA/symmetric.decl
- include/BCR_CPP_LA/symmetric.h

4.48 valerg Struct Reference

Pair of property value and energy.

#include <typedefs.hh>

Public Attributes

- double property
- double penalty
- double energy
- bool property_computed
- bool energy_computed

4.48.1 Member Data Documentation

4.48.1.1 double valerg::energy

4.48.1.2 bool valerg::energy_computed

4.48.1.3 double valerg::penalty

4.48.1.4 double valerg::property

4.48.1.5 bool valerg::property_computed

The documentation for this struct was generated from the following file:

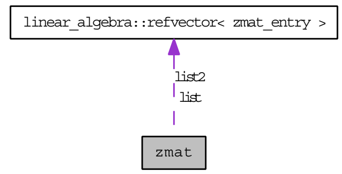
- typedefs.hh

4.49 zmat Class Reference

Class for Z-matrices. Just a wrapper class around a vector of entries.

#include <zmat.hh>

Collaboration diagram for zmat:



Public Member Functions

- zmat & operator= (const zmat &a)
  Assignment operator.
- bool operator== (const zmat &a) const
  Comparison operator.
- zmat & set_val (int i, int j, double val)
  Set a value.
- zmat & add_val (int i, int j, double val)
  Add to a value.
- zmat & add_entry (const zmat_entry &e)
  Add an entry to the Z-matrix.
- void output (ostream &out) const
  Output the z-matrix definition.
- zmat & add_increment (long i, long j, double a)
  Add another value to entry i in position j.
- zmat & add_zmat (const zmat &B)
  Combine two Z-matrices. B.offset must be zero!
- void set_opt_val (long i, long j, bool val)
  Set the optimization flag for zmat_entry i in position j.
- zmat & add_zmat (const zmat &B, const zmat_connector &e)
  Combine two Z-matrices. B.offset must be zero!
- zmat & concat_zmat (const zmat&B)
  Concat two Z-matrices with different offsets.
- zmat & correct_zmat (const zmat_entry &x, const long newoff)
- long count_constants () const
  Count the number of constants.

- long count_variables () const
  Count the number of variables.

- void set_constants_variables (const refvector< double > &consts, const refvector< double > &vars)
  Set the constants and variables of a Z-matrix from two refvectors.

- stringstream & zmat_to_string (long N, stringstream &output) const
  Output a Z-matrix to a stringstream.

- void update_variables (const zmat &B)
  Update the variables in the matrix without touching connectivity or increments.

Constructors

- zmat ()
  Default constructor.

- zmat (const long o)
  Constructor which uses a non-zero offset.

- zmat (const zmat &a)
  Copy constructor.

- zmat (const refvector< zmat_entry > &z)
  Construction from a vector of zmat_entries.

- zmat (stringstream &s)
  Construct from a stringstream.

Public Attributes

- const refvector< zmat_entry > & list
  read-only access to list2

- const long & offset_r
  read-only access to offset.

Private Attributes

- refvector< zmat_entry > list2
  List of zmat_entries.

- long offset

    Auxiliary offset for connecting matrices.

4.49.1 Detailed Description

The Z-matrix may have negative connectivity entries in the first upper triangle of definition. This is for the purpose of combining Z-matrices. These entries are ignored when the zmat is output. The zmat entries are always checked for consistency upon construction.

4.49.2 Constructor & Destructor Documentation

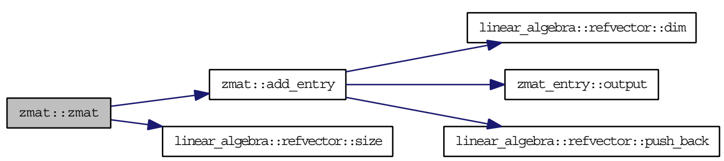
4.49.2.1 zmat::zmat ()[inline]

4.49.2.2 zmat::zmat (const long *o*)[inline]

4.49.2.3 zmat::zmat (const zmat & *a*)[inline]

4.49.2.4 zmat::zmat (const refvector< zmat_entry > & *z*)[inline]

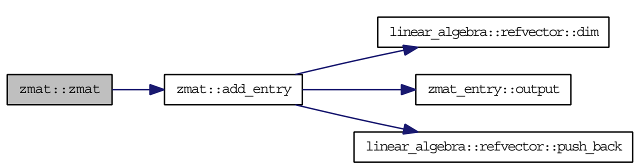
Here is the call graph for this function:



4.49.2.5 zmat::zmat (stringstream & *s*)[inline]

The format goes (zmat_entry ... zmat_entry)

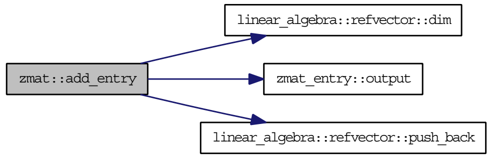
Here is the call graph for this function:



4.49.3 Member Function Documentation

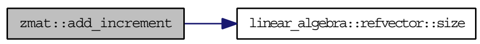
4.49.3.1 zmat& zmat::add_entry (const zmat_entry & *e*)[inline]

Here is the call graph for this function:



4.49.3.2 zmat& zmat::add_increment (long i, long j, double a)[inline]

Here is the call graph for this function:



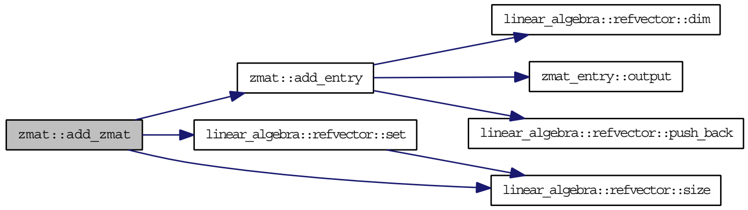
4.49.3.3 zmat& zmat::add_val (int i, int j, double val)[inline]

Here is the call graph for this function:

4.49.3.4 zmat& zmat::add_zmat (const zmat & B, const zmat_connector & e)[inline]

The first entry is defined to be exactly e with the corresponding Name, i.e., [B.list[0].Name -1 e.connect[0] -2 e.connect[1] -3 e.connect[2]].

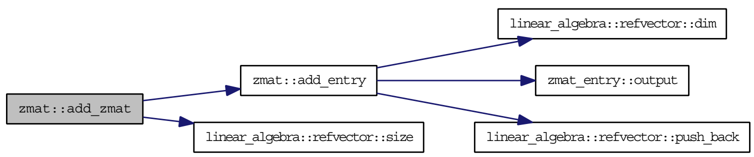
Here is the call graph for this function:



4.49.3.5 zmat& zmat::add_zmat (const zmat & *B*)[inline]

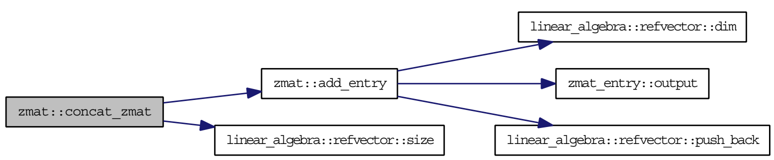
It is assumed that all B entries refer to the last three entries of (∗this) or B itself.

Here is the call graph for this function:



4.49.3.6 zmat& zmat::concat_zmat (const zmat & *B*)[inline]

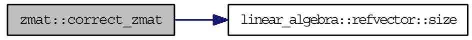
Here is the call graph for this function:



4.49.3.7 zmat& zmat::correct_zmat (const zmat_entry & *x*, const long *newoff*)[inline]

This has the effect of inserting blank space between the original offset matrix and the new matrix as well as updating the connectors.

Here is the call graph for this function:



4.49.3.8 long zmat::count_constants () const[inline]

4.49.3.9 long zmat::count_variables () const[inline]

4.49.3.10 zmat& zmat::operator= (const zmat & *a*)[inline]

4.49.3.11 bool zmat::operator== (const zmat & *a*) const[inline]

4.49.3.12 void zmat::output (ostream & *out*) const[inline]

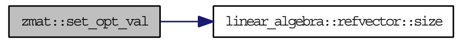
Here is the call graph for this function:

4.49.3.13 void zmat::set_constants_variables (const refvector< double > & *consts*, const refvector< double > & *vars*)[inline]

Here is the call graph for this function:

4.49.3.14 void zmat::set_opt_val (long *i*, long *j*, bool *val*)[inline]

Here is the call graph for this function:

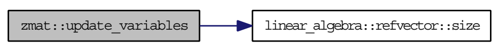


4.49.3.15 zmat& zmat::set_val (int i, int j, double val)[inline]

Here is the call graph for this function:

4.49.3.16 void zmat::update_variables (const zmat & B)[inline]

Here is the call graph for this function:

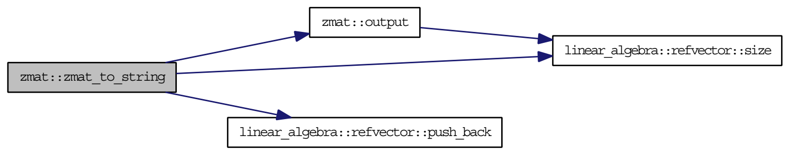


4.49.3.17 stringstream& zmat::zmat_to_string (long N, stringstream & *output*) const[inline]

Parameters:

Nsignifies the conformation of the combinatorial product of options.

Here is the call graph for this function:



4.49.4 Member Data Documentation

4.49.4.1 const refvector<zmat_entry>& zmat::list

4.49.4.2 refvector<zmat_entry> zmat::list2[private]

4.49.4.3 long zmat::offset[private]

4.49.4.4 const long& zmat::offset_r

The documentation for this class was generated from the following file:

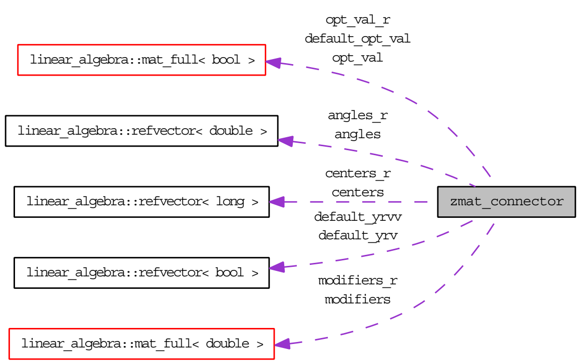
- zmat.hh

4.50 zmat_connector Class Reference

Class of connectors between z-matrices.

#include <zmat.hh>

Collaboration diagram for zmat_connector:



Public Member Functions

- zmat_connector & operator= (const zmat_connector &A)

Assignment operator.

- bool operator== (const zmat_connector &A) const
  Comparison operator.

- stringstream & output (stringstream &s) const
  Output the connector to a stringstream.

- void set_opt_val (long i, long j, bool v)
  Set the optimization flag .

- void add_angle (double i)
  Add an angle to the.

Constructors

- zmat_connector ()
  Default constructor.

- zmat_connector (const zmat_entry &a)
  Construct from a zmat_entry.

- zmat_connector (const zmat_entry &a, const mat_full< bool > &nopt_val)
  Construct from a zmat_entry and a set of optimization flags.

- zmat_connector (stringstream &s)
  Construct from an input string stream.

- zmat_connector (const zmat_connector &A)
  Construct from another zmat_connector.

Static Public Member Functions

- static zmat_connector & update_connector (const zmat_connector &a, const zmat_connector &e, const long add, zmat_connector &x)
  Update a connector to fit after two Z-matrices have been combined.

- static zmat_connector & update_connector (const zmat_connector &a, const long add, zmat_connector &x)
  Update a connector to fit after two Z-matrices have been combined.

Public Attributes

- const refvector< long > & centers_r
  Translation vector of -3, -2, -1 to alternate centers. (READ ONLY).
- const mat_full< double > & modifiers_r
  Modification matrix. Depending on the use of each center modifications may be different. (READ ONLY).
- const mat_full< bool > & opt_val_r
  Matrix of flags to set optimization.
- const refvector< double > & angles_r
  deg/*angle for conformational purposes. (READ ONLY)*

Static Public Attributes

- static const bool default_y [3] = {false, false, false}
- static const refvector< bool > default_yrv
- static const refvector< bool > default_yrvv [3]
- static const refvector< refvector< bool > > default_yrvrv
- static const mat_full< bool > default_opt_val

Private Attributes

- refvector< long > centers
  Translation vector of -3, -2, -1 to alternate centers.
- mat_full< double > modifiers
  Modification matrix. Depending on the use of each center modifications may be different.
- mat_full< bool > opt_val
  Matrix of flags to set optimization.
- refvector< double > angles
  deg/*angle for conformational purposes.*

4.50.1 Detailed Description

A zmat_connector has three distinct references:

- –6 to –4 refer to the original connector of the following group. (default)
- –3 to –1 refer to values of the connector passed down to the current group.
- >–1 refers to values in the local Z-matrix.

4.50.2 Constructor & Destructor Documentation

4.50.2.1 zmat_connector::zmat_connector ()

4.50.2.2 zmat_connector::zmat_connector (const zmat_entry & *a*)

4.50.2.3 zmat_connector::zmat_connector (const zmat_entry & *a*, const mat_full< bool > & *nopt_val*)

4.50.2.4 zmat_connector::zmat_connector (stringstream & *s*)

The connector information is generated from:(C1,C2,C3)
(V11,V12,V13)
(V21,V22,V23)
(V31,V32,V33)
(O11,O12,O13)
(O11,O12,O13)
(O11,O12,O13)
(Oa,Ob)
Ci are integers referencing connectors, Vij are variables depending on the position Ci in a zmat_entry Oij is a flag 0/1 for optimization of this variable

4.50.2.5 zmat_connector::zmat_connector (const zmat_connector & *A*)

4.50.3 Member Function Documentation

4.50.3.1 void zmat_connector::add_angle (double *i*

4.50.3.2 zmat_connector & zmat_connector::operator= (const zmat_connector & *A*)

4.50.3.3 bool zmat_connector::operator== (const zmat_connector & *A*) const

4.50.3.4 stringstream & zmat_connector::output (stringstream & s) const

4.50.3.5 void zmat_connector::set_opt_val (long i, long j, bool v)

4.50.3.6 zmat_connector & zmat_connector::update_connector (const zmat_connector & a, const long add, zmat_connector & x)[static]

4.50.3.7 zmat_connector & zmat_connector::update_connector (const zmat_connector & a, const zmat_connector & e, const long add, zmat_connector & x)[static]

4.50.4 Member Data Documentation

4.50.4.1 refvector<double> zmat_connector::angles[private]

4.50.4.2 const refvector<double>& zmat_connector::angles_r

4.50.4.3 refvector<long> zmat_connector::centers[private]

4.50.4.4 const refvector<long>& zmat_connector::centers_r

4.50.4.5 const mat_full< bool > zmat_connector::default_opt_val[static]

4.50.4.6 const bool zmat_connector::default_y = {false, false, false}[static]

4.50.4.7 const refvector< bool > zmat_connector::default_yrv[static]

4.50.4.8 const refvector< refvector< bool > > zmat_connector::default_yrvrv[static]

4.50.4.9 const refvector< bool > zmat_connector::default_yrvv[static]

Initial value:

refvector<bool>(3),
zmat_connector::default_yrv,
zmat_connector::default_yrv

4.50.4.10 mat_full<double> zmat_connector::modifiers[private]

4.50.4.11 const mat_full<double>& zmat_connector::modifiers_r

4.50.4.12 mat_full<bool> zmat_connector::opt_val[private]

4.50.4.13 mat_full<bool>& zmat_connector::opt_val_r

The documentation for this class was generated from the following files:

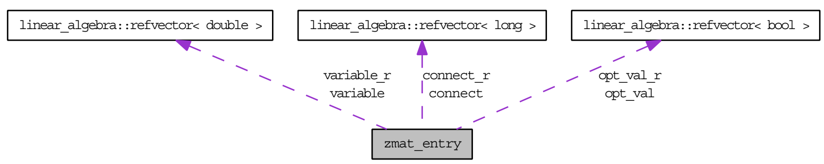
- zmat.hh
- zmat.cc

4.51 zmat_entry Class Reference

Class of Z-matrix entries.

#include <zmat.hh>

Collaboration diagram for zmat_entry:



Public Member Functions

- zmat_entry & operator= (const zmat_entry &a)
  Copy assignment.
- bool operator== (const zmat_entry &a) const
  Comparison operator.
- stringstream & output (stringstream &s) const
  Output the entry to a stringstream.
- void update_variables (const zmat_entry &b)
  Update the variables in the zmat_entry without touching connectivity or increments.

Constructors

- zmat_entry ()
  Default constructor. zmat_entry (const zmat_entry &a)
  Copy constructor. zmat_entry (const string &N)
  Named default construction.
- zmat_entry (const string &N, const double ∗v, const int ∗c)
  Construction with full initialization.

- zmat_entry (const string &N, const double *v)
  Construction with variable initialization.
- zmat_entry (const string &N, const refvector< double > &v)
  Construction with variable initialization.
- zmat_entry (const string &N, const refvector< double > &v, const refvector< long > &c)
  Construction from explicit values.
- zmat_entry (stringstream &s)
  Construction from a stringstream.

Public Attributes

- const string & Name_r
  Atom name.
- const refvector< double > & variable_r
  Array of variables.
- const refvector< refvector< double > > & increment_r
  Potential alternate values for each variable.
- const refvector< long > & connect_r
  Connectivity data.
- const refvector< bool > & opt_val_r
  Optimization flags for each variable.

Static Public Attributes

- static const bool default_opt_val [3] = {false,false, false}

Private Attributes

- string Name
  Atom name.
- refvector< double > variable
  Array of variables.

- refvector< refvector< double > > increment
  Potential alternate values for each variable.
- refvector< long > connect
  Connectivity data.
- refvector< bool > opt_val
  Optimization flags for each variable.

Friends

- class zmat

4.51.1 Constructor & Destructor Documentation

4.51.1.1 zmat_entry::zmat_entry ()

4.51.1.2 zmat_entry::zmat_entry (const zmat_entry & a)

4.51.1.3 zmat_entry::zmat_entry (const string & N)

4.51.1.4 zmat_entry::zmat_entry (const string & N, const double $*$ v, const int $*$ c)

4.51.1.5 zmat_entry::zmat_entry (const string & N, const double $*$ v)

4.51.1.6 zmat_entry::zmat_entry (const string & N, const refvector< double > & v)

4.51.1.7 zmat_entry::zmat_entry (const string & N, const refvector< double > & v, const refvector< long > & textit c)

4.51.1.8 zmat_entry::zmat_entry (stringstream & s)

Each entry in the string has the followin form: (Name, Integer(f), ength(double,...,double), Integer(f), Angle(double,...,double), Integer(f), Dihedral(double,...,double))

- Name is a string.
- f is a flag of 0 or 1. The inner parentheses are only needed if specific values are given.

4.51.2 Member Function Documentation

4.51.2.1 zmat_entry & zmat_entry::operator= (const zmat_entry & a)

4.51.2.2 bool zmat_entry::operator== (const zmat_entry & *a*) const

4.51.2.3 stringstream & zmat_entry::output (stringstream & *s*) const

4.51.2.4 void zmat_entry::update_variables (const zmat_entry & *b*)

4.51.3 Friends and Related Function Documentation

4.51.3.1 friend class zmat[friend]

4.51.4 Member Data Documentation

4.51.4.1 refvector<long> zmat_entry::connect[private]

4.51.4.2 const refvector<long>& zmat_entry::connect_r

4.51.4.3 const bool zmat_entry::default_opt_val = {false,false, false}[static]

4.51.4.4 refvector<refvector<double> > zmat_entry::increment[private]

4.51.4.5 const refvector<refvector<double> >& zmat_entry::increment_r

4.51.4.6 string zmat_entry::Name[private]

4.51.4.7 const string& zmat_entry::Name_r

4.51.4.8 refvector<bool> zmat_entry::opt_val[private]

4.51.4.9 refvector<bool>& zmat_entry::opt_val_r

4.51.4.10 refvector<double> zmat_entry::variable[private]

4.51.4.11 const refvector<double>& zmat_entry::variable_r

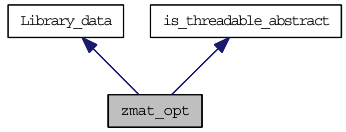
The documentation for this class was generated from the following files:

- zmat.hh
- zmat.cc

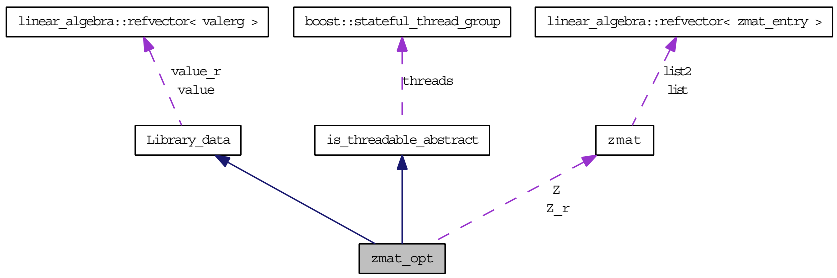
4.52 zmat_opt Class Reference

#include <zmat_opt.hh>

Inheritance diagram for zmat_opt:



Collaboration diagram for zmat_opt:



Public Member Functions

- const zmat & operator= (const zmat &A)
  Assignment operator for a zmat. valerg compute_property ulong i) const
  Compute the property. (Here energy is important).
- valerg compute_property (ulong i, zmat &A) const
  Compute the property. (Here energy is important).
- valerg compute_energy (ulong i) const
  Compute the energy. (Here energy is important).
- valerg compute_energy (ulong i, zmat &A) const
  Compute the energy. (Here energy is important).
- ulong get_space_size () const
  Compute the size of the optimization space.
- ulong get_bits () const
  Compute the number of bits to address the optimization space.
- bool pre_opt (ulong N) const
  Find a converged starting geometry.

Constructors

- zmat_opt ()
  Default constructor. zmat_opt (const zmat_opt &A)
  Copy constructor. zmat_opt (const zmat &A)
  Initialize with a zmatrix.

Public Attributes

- const zmat & Z_r
  Read-only access to Z.

- bool compute_property_flag

Private Attributes

- zma

4.52.1 Detailed Description

zmat_opt describes a threadable Library using class zmat. It has two associated properties as expressed by compute_energy(ulong i) and compute_property(ulong i). Set compute_property_flag in order to compute the property and energy of a given conformation.

4.52.2 Constructor & Destructor Documentation

4.52.2.1 zmat_opt::zmat_opt ()

4.52.2.2 zmat_opt::zmat_opt (const zmat_opt & *A*)

4.52.2.3 zmat_opt::zmat_opt (const zmat & *A*)

4.52.3 Member Function Documentation

4.52.3.1 valerg zmat_opt::compute_energy (ulong *i*, zmat & *A*) const

This method is necessary for initialisation purposes.

4.52.3.2 valerg zmat_opt::compute_energy (ulong i) const

4.52.3.3 valerg zmat_opt::compute_property (ulong i, zmat & A) const

This method is necessary for initialisation purposes.

4.52.3.4 valerg zmat_opt::compute_property (ulong i) const[virtual]

Implements Library_data.

4.52.3.5 ulong zmat_opt::get_bits () const[virtual]

Implements Library_data.

4.52.3.6 ulong zmat_opt::get_space_size () const[virtual]

Implements Library_data.

4.52.3.7 const zmat & zmat_opt::operator= (const zmat & A)

4.52.3.8 bool zmat_opt::pre_opt (ulong N) const

4.52.4 Member Data Documentation

4.52.4.1 bool zmat_opt::compute_property_flag

Changes the behavior of compute_property.

- true: compute property and energy
- false: compute energy only.

4.52.4.2 zmat zmat_opt::Z[mutable, private]

4.52.4.3 const zmat& zmat_opt::Z_r

The documentation for this class was generated from the following files:

- zmat_opt.hh
- zmat_opt.cc

5. Discrete Optimization in Chemical Space File Documentation

5.1 binary_entropic.hh File Reference

Definition and Implementation of the binary_entropic class.

#include <typedefs.hh>
#include <concepts.hh>
#include <iostream>
#include <optimizeabstract.h>
#include <cmath>
#include <BCR_CPP_LA/linear_algebra.h>
#include <entropic_aux.hh>

Namespaces

- namespace std

Classes

- class binary_entropic< C >
  Class for enhanced sampling using an entropic measure of coverage.

5.1.1 Detailed Description

5.2 binary_line_search.hh File Reference

implementation of a line search optimization

#include <concepts.hh>
#include <optimizeabstract.h>

Classes

- class binary_line_search< C >
  Search a pruned Library using a linesearch algorithm.

5.2.1 Detailed Description

5.3 binary_opt.cc File Reference

#include <BCR_CPP_LA/refcount.h>
#include <chemgroup.hh>
#include <chem_opt.hh>
#include <Library_data.hh>
#include <zmat.hh>
#include <sstream>
#include <fstream>
#include <iostream>
#include <cmath>
#include <noprune.h>
#include <simpleprune.h>
#include <simpleprune.cc>
#include <optimizeabstract.h>
#include <binary_line_search.hh>
#include <binarysteepestdescent.hh>
#include <binarygdmc.hh>
#include <genbase-l-s.hh>
#include <genbase-grad-ls.hh>
#include <boost/concept_check.hpp>
#include <concepts.hh>
#include <reordergeneralbase.hh>
#include <generalbaseiterator.hh>
#include <genbasegdmc.hh>
#include <gen_base_entropic.hh>
#include <binary_entropic.hh>

Functions

- valerg calc_property (const zmat &A, const string &out, const string &id, zmat &returnA)
  Setup the external computations and execute them.
- template<class X> void print (const X &D)
- template<class X> ulong run (const X &D)
- int main (int argc, char *argv[])
  Main execution routine. Handles command-line input and executes optimization.

5.3.1 Function Documentation

5.3.1.1 valerg calc_property (const zmat & *A*, const string & *out*, const string & *id*, zmat & *returnA*)

5.3.1.2 int main (int *argc*, char ∗ *argv*[ ])

5.3.1.3 template<class X> void print (const X & *D*)

5.3.1.4 template<class X> ulong run (const X & *D*)

5.4 binarygdmc.hh File Reference

Implementation of binary GDMC.

#include <typedefs.hh>
#include <concepts.hh>
#include <iostream>
#include <optimizeabstract.h>

Classes

- class binary_gdmc< C >
  Gradient-directed Monte Carlo performed on a hypercube.

5.4.1 Detailed Description

5.5 binarysteepestdescent.hh File Reference

Implements the binary_steepest_descent optimization class.

#include <optimizeabstract.h>
#include <concepts.hh>
#include <BCR_CPP_LA/refcount.h>
#include <has_gradients_hessian_data.hh>

Classes

- class binary_steepest_descent< C >
  Searches a Lilbrary for an optimum using a steepest descent method.

5.5.1 Detailed Description

5.6 chem_opt.cc File Reference

Implementation of chemical optimization class.

#include <BCR_CPP_LA/refcount.h>
#include <chemgroup.hh>
#include <zmat_opt.hh>
#include <chem_opt.hh>
#include <cmath>
#include <binary_line_search.hh>

Defines

- #define CLASS template <class P> chem_opt

5.6.1 Detailed Description

5.6.2 Define Documentation

5.6.2.1 #define CLASS template <class P> chem_opt

5.7 chem_opt.hh File Reference

#include <BCR_CPP_LA/refcount.h>
#include <typedefs.hh>
#include <chemgroup.hh>
#include <zmat.hh>
#include <Library_data.hh>
#include <isthreadableabstract.h>
#include <concepts.hh>
#include <boost/concept_check.hpp>

Classes

- chem_opt
  Chemical optimization class.

5.8 chemgroup.cc File Reference

Implementation file of class ChemGroup.

#include <BCR_CPP_LA/refcount.h>
#include <chemgroup.hh> %

#include <zmat.hh>
#include <string>
#include <sstream>
#include <iostream>
#include <cmath>

Functions

- bool filter (const ChemGroup &A)
- refvector< unsigned long > & sort_descending (const refvector)< double > &E, unsigned long start, unsigned long end, refvector< unsigned long > &index)
  Do a qsort on E with immediate storage back into index. (perfectly parallelizable).
- refvector< unsigned long > & sort_descending (const refvector< double > &E, refvector< unsigned long > &index)
- static void spaces (long n)
  outputs a number of spaces

5.8.1 Detailed Description

5.8.2 Function Documentation

5.8.2.1 bool filter (const ChemGroup & *A*)

5.8.2.2 refvector<unsigned long>& sort_descending (const refvector< double > & *E*, refvector< unsigned long > & *index*)

5.8.2.3 refvector<unsigned long>& sort_descending (const refvector< double > & *E*, unsigned long *start*, unsigned long *end*, refvector< unsigned long > & *index*)

5.8.2.4 static void spaces (long *n*)[static]

5.9 chemgroup.hh File Reference

Header file for ChemGroup class.

#include <BCR_CPP_LA/refcount.decl>
#include <zmat.hh>
#include <iostream>
#include <sstream>
#include <typedefs.hh>
#include <chemident.hh>

Classes

- class ChemGroup
  This class describes a group of substitution sites and their respective substitution options as well as computation history.

Typedefs

- typedef unsigned long ulong

5.9.1 Detailed Description

5.9.2 Typedef Documentation

5.9.2.1 typedef unsigned long ulong

5.10 chemident.cc File Reference

Implementation file of class ChemIdent.

#include <BCR_CPP_LA/refcount.h>
#include <chemident.hh>
#include <zmat.hh>
#include <zmat_opt.hh>
#include <string>
#include <sstream>
#include <iostream>
#include <cmath>

Functions

- static void (long n)

Variables

- static const double zeros [3] = {0.0,0.0,0.0}
- static const int conn [3] = {-6,-5,-4}
- static const zmat_connector default_return_connector (zmat_entry("_", zeros, conn))

5.10.1 Detailed Description

5.10.2 Function Documentation

5.10.2.1 static void spaces (long *n*)[static]

5.10.3 Variable Documentation

5.10.3.1 const int conn[3] = {-6,-5,-4}[static]

5.10.3.2 const zmat_connector default_return_connector(zmat_entry("_", zeros, conn))[static]

5.10.3.3 const double zeros[3] = {0.0,0.0,0.0}[static]

5.11 chemident.hh File Reference

Header file for ChemIdent class.

#include <BCR_CPP_LA/linear_algebra.h>
#include <zmat.hh>
#include <zmat_opt.hh>
#include <iostream>
#include <sstream>
#include <typedefs.hh>

Classes

- class ChemIdent
  This class describes a group of substitution sites on a Z-matrix and their respective substitution options.

5.11.1 Detailed Description

5.12 concepts.hh File Reference

#include <boost/concept_check.hpp>
#include <BCR_CPP_LA/refcount.decl>
#include <BCR_CPP_LA/mat_sym_full.decl>
#include <typedefs.hh>
#include <boost/thread/shared_mutex.hpp>
#include <stateful_thread_group.hh>
#include <string>

Namespaces

- namespace Concepts

Classes

- class Concepts::Library< X >
  Concept defines an addressable set of values.
- class Concepts::base_iterator< X >
  Concept defines an iterator for a class of bases.
- class Concepts::pruner< X >
  Concept defines an addressable set of values.
- class Concepts::has_gradients< X >
  Concept defines a class that has gradients.
- class Concepts::has_hessians< X >
  Concept defines a class that has hessians.
- class Concepts::has_Name< X >
  Concept defines a class that has a private string Name.
- class Concepts::is_threadable< X >
  Concept defines a class that is threadable.
- class Concepts::has_optimize< X >
  Concept defines a class that has an optimize member.
- class Concepts::has_stacksize< X >
  Concept defines a class that has a stack member.

Functions

- template<typename T> void Concepts::same_type (T const &, T const &)
  Dummy function to determine type equality.

5.12.1 Detailed Description

This file contains the assembled concepts used in the project Discrete Molecular Optimization.

5.13 entropic_aux.cc File Reference

#include <iostream>
#include <cmath>
#include <BCR_CPP_LA/linear_algebra.h>

Functions

- void linsolve_cg (const mat_sym_full< double > &J, const refvector< double > &G, refvector< double > &X)
  Conjugate gradients for linear solver.
- refvector< double > & set_gradient (const refvector< long > &b, const mat_full< double > &H, const refvector< double > &X, refvector< double > &G)
  We ignore infinities in the derivatives.
- mat_sym_full< double > & set_hessian (const refvector< long > &b, const mat_full< double > &H, const refvector< double > &X, mat_sym_full< double > &J)
  We ignore infinities in the derivatives.
- ulong maximize_entropic_distance (const refvector< ulong > &A, const refvector< long > &b)
  Maixmize the entropic distance as declared in set_gradient() using Newton-Raphson.

5.13.1 Function Documentation

5.13.1.1 const mat_sym_full< double > & J, const refvector< double > & G, refvector< double > & X)

5.13.1.2 ulong maximize_entropic_distance (const refvector< ulong > & A, const refvector< long > & b)

5.13.1.3 refvector<double>& set_gradient (const refvector< long > & b, const mat_full< double > & H, const refvector< double > & X, refvector< double > & G)

Compute the gradient of the distance function $f = \sum_i \ln \sqrt{\sum_j \sin(x_j - s_j^{(i)}\pi/n_j)^2}$ where n_j is the number of digits in component j.

5.13.1.4 mat_sym_full<double>& set_hessian (const refvector< long > & b, const mat_full< double > & H, const refvector< double > & X, mat_sym_full< double > & J)

Compute the hessian of the distance function $f = \sum_i \ln \sqrt{\sum_j \sin(x_j - s_j^{(i)}\pi/n_j)^2}$ where n_j is the number of digits in component j.

5.14 entropic_aux.hh File Reference

Auxiliary functions for ∗_entropic classes.

#include <iostream>
#include <cmath>
#include <BCR_CPP_LA/linear_algebra.h>

Functions

- void linsolve_cg (const mat_sym_full< double > &J, const refvector< double > &G, refvector< double > &X)
  Conjugate gradients for linear solver.
- refvector< double > & set_gradient (const refvector< long > &b, const mat_full< double > &H, const refvector< double > &X, refvector< double > &G)
  We ignore infinities in the derivatives.
- mat_sym_full< double > & set_hessian (const refvector< long > &b, const mat_full< double > &H, const refvector< double > &X, mat_sym_full< double > &J)
  We ignore infinities in the derivatives.
- ulong maximize_entropic_distance (const refvector< ulong > &A, const refvector< long > &b)
  Maixmize the entropic distance as declared in set_gradient() using Newton-Raphson.

5.14.1 Detailed Description

5.14.2 Function Documentation

5.14.2.1 void linsolve_cg (const mat_sym_full< double > & J, const refvector< double > & G, refvector< double > & X)

5.14.2.2 ulong maximize_entropic_distance (const refvector< ulong > & A, const refvector< long > & b)

5.14.2.3 refvector<double>& set_gradient (const refvector< long > & b, const mat_full< double > & H, const refvector< double > & X, refvector< double > & G)

Compute the gradient of the distance function $f = \sum_i \ln \sqrt{\sum_j \sin(x_j - s_j^{(i)}\pi/n_j)^2}$ where n_j is the number of digits in component j.

5.14.2.4 mat_sym_full<double>& set_hessian (const refvector< long > & b, const mat_full< double > & H, const refvector< double > & X, mat_sym_full< double > & J)

Compute the hessian of the distance function $f = \sum_i \ln \sqrt{\sum_j \sin(x_j - s_j^{(i)}\pi/n_j)^2}$ where n_j is the number of digits in component j.

5.15 gen_base_entropic.hh File Reference

#include <typedefs.hh>
#include <concepts.hh>
#include <iostream>
#include <optimizeabstract.h>
#include <cmath>
#include <BCR_CPP_LA/linear_algebra.h>
#include <entropic_aux.hh>

Classes

- class gen_base_entropic< C >
  Meta-Optimize by generating maximally distant starting configurations.

5.16 genbase-grad-ls.hh File Reference

Line search optimization with general bases.

#include <typedefs.hh>
#include <concepts.hh>
#include <iostream>
#include <optimizeabstract.h>
#include <BCR_CPP_LA/refcount.h>

Classes

- class gen_base_grad_LS< C, B >
  Line search optimization class using general bases.

5.16.1 Detailed Description

5.17 genbase-l-s.hh File Reference

Line search optimization with general bases.

#include <typedefs.hh>
#include <concepts.hh>
#include <iostream>
#include <optimizeabstract.h>
#include <BCR_CPP_LA/refcount.h>

Classes

- class gen_base_LS< C, B >
- *L*ine search optimization class using general bases.

5.17.1 Detailed Description

5.18 generalbaseiterator.cc File Reference

TODO: Throw exceptions.

#include <chem_opt.hh>

#include <chemident.hh>

#include <generalbaseiterator.hh>

5.18.1 Detailed Description

5.19 generalbaseiterator.hh File Reference

#include <concepts.hh>

#include <BCR_CPP_LA/refcount.h>

#include <typedefs.hh>

#include <chemident.hh>

Classes

- class general_base_iterator< X >
  Iterator over the potential bases.

5.19.1 Detailed Description

5.20 has_gradients_hessian_data.decl File Reference

#include <BCR_CPP_LA/refcount.decl>
#include <BCR_CPP_LA/mat_sym_full.decl>

#include "typedefs.hh"
#include <string>
#include <boost/thread.hpp>
#include <stateful_thread_group.hh>
#include "concepts.hh"

Classes

- class has_gradients_data< X >
  Abstract class for discrete optimizations.

- class has_hessians_data< X >
  Abstract class for discrete optimizations.

5.21 has_gradients_hessian_data.hh File Reference

#include <has_gradients_hessian_data.decl>

#include <BCR_CPP_LA/refcount.h>
#include <BCR_CPP_LA/mat_sym_full.h>
#include <typedefs.hh>
#include <stdexcept>

5.22 include/BCR_CPP_LA/ABmBA.hh File Reference

#include <linear_algebra.h>

Namespaces

- namespace linear_algebra

Functions

- template<class TN> mat_sym_full< TN > & linear_algebra::ABmBA (const mat_sym_full< TN > A, const mat_asym_full< TN > B, mat_sym_full< TN > &r)
  Computes the commutator AB-BA.
- template<class TN> mat_asym_full< TN > linear_algebra::ABmBA (const mat_sym_full< TN > A, const mat_asym_full< TN > B)
  Computes the commutator AB-BA.
- template<class TN> mat_asym_full< TN > & linear_algebra::ABmBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B, mat_asym_full< TN > &r)
  Computes the commutator AB-BA.
- template<class TN> mat_asym_full< TN > linear_algebra::ABmBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B)
  Computes the commutator AB-BA.
- template<class TN> mat_asym_full< TN > & linear_algebra::ABmBA (const mat_asym_full< TN > &A, const mat_asym_full< TN > &B, mat_asym_full< TN > &r)
  Computes the commutator AB-BA.
- template<class TN> mat_asym_full< TN > linear_algebra::ABmBA (const mat_asym_full< TN > &A, const mat_asym_full< TN > &B)
  Computes the commutator AB-BA.

5.22.1 Detailed Description

5.23 include/BCR_CPP_LA/ABpBA.hh File Reference

#include <linear_algebra.h>

Namespaces

- namespace linear_algebra

Functions

- template<class TN> mat_asym_full< TN > & linear_algebra::ABpBA (const mat_sym_full< TN > &A, const mat_asym_full< TN > &B, mat_asym_full< TN > &r)
  Computes the anti-commutator AB+BA.
- template<class TN> mat_asym_full< TN > linear_algebra::ABpBA (const mat_sym_full< TN > &A, const mat_asym_full< TN > &B)
  Computes the anti-commutator AB+BA.
- template<class TN> mat_asym_full< TN > linear_algebra::ABpBA (const mat_full< TN > &A, const mat_asym_full< TN > &B, mat_asym_full< TN > &r)
  Computes the anti-commutator AB+BA.
- template<class TN> mat_asym_full< TN > linear_algebra::ABpBA (const mat_full< TN > A, const mat_asym_full< TN > B)
  Computes the anti-commutator AB+BA.
- template<class TN> mat_asym_full< TN > & linear_algebra::ABpBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B, mat_asym_full< TN > &r)
  Computes the anti-commutator AB+BA.
- template<class TN> mat_asym_full< TN > linear_algebra::ABpBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B)
  Computes the anti-commutator AB+BA.
- template<class TN> mat_asym_full< TN > & linear_algebra::ABpBA (const mat_sym_full< TN > A, const mat_asym_sparse< TN > B, mat_asym_full< TN > &r)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > linear_algebra::ABpBA (const mat_sym_full< TN > A, const mat_asym_sparse< TN > B)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > & linear_algebra::ABpBA (const mat_asym_sparse< TN > A, const mat_sym_full< TN > B, mat_asym_full< TN > &r)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > linear_algebra::ABpBA (const mat_asym_sparse< TN > A, const mat_sym_full< TN > B)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_sym_full< TN > & linear_algebra::ABpBA (const mat_sym_full< TN > A, mat_sym_full< TN > B, mat_sym_full< TN > &r, mat_sym_full< TN > &I, mat_full< TN > &I2)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_sym_full< TN > & linear_algebra::ABpBA (const mat_sym_full< TN > A, mat_sym_full< TN > B, mat_sym_full< TN > &r)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_sym_full< TN > & linear_algebra::ABpBA (const mat_sym_full< TN > &A, const mat_sym_full< TN > &B, mat_sym_full< TN > &r, mat_sym_full< TN > &I)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_sym_full< TN > linear_algebra::ABpBA (const mat_sym_full< TN > &A, const mat_sym_full< TN > &B)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_sym_full< TN > & linear_algebra::ABpBA (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_sym_full< TN > &r, mat_full< TN > &I2)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_sym_full< TN > & linear_algebra::ABpBA (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_sym_full< TN > &r)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_sym_full< TN > linear_algebra::ABpBA (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B)
  Computes the anti-commutator AB+BA.

- template<class TN> mat_asym_full< TN > & linear_algebra::ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_full< TN > &r, mat_full< TN > &I2)
    Computes the anti-commutator AB+BA.
- template<class TN> mat_asym_sparse< TN > & linear_algebra::ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_sparse< TN > &r, mat_sparse< TN > &I2)
    Computes the anti-commutator AB+BA.
- template<class TN> mat_asym_full< TN > & linear_algebra::ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_full< TN > &r, mat_sparse< TN > &I2)
    Computes the anti-commutator AB+BA.
- template<class TN> mat_asym_full< TN > & linear_algebra::ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_full< TN > &r)
    Computes the anti-commutator AB+BA.
- template<class TN> mat_asym_sparse< TN > & linear_algebra::ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_sparse< TN > &r)
    Computes the anti-commutator AB+BA.
- template<class TN> mat_asym_full< TN > linear_algebra::ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B)
    Computes the anti-commutator AB+BA.
- template<class T> mat_sym_full< T > linear_algebra::ABpBA (const mat_asym_full< T > &A, const mat_asym_full< T > &B)
    Commutes the anti-commutator AB+BA. Lame and lazy version.

5.23.1 Detailed Description

5.24 include/BCR_CPP_LA/asymmetric.decl File Reference

Provide declarations for antisymmetric full matrices.

#include <vector>
#include <linear_algebra.decl>
#include <matrix.decl>
#include <square.decl>

Namespaces

- namespace linear_algebra

Classes

- class linear_algebra::asymmetric< TN >
  Class of antisymmetric matrices.

5.24.1 Detailed Description

Todo

throw exceptions.

5.25 include/BCR_CPP_LA/asymmetric.h File Reference

Provide manipulations for antisymmetric full matrices.

#include <vector>
#include <square.h>
#include <asymmetric.decl>

Namespaces

- namespace linear_algebra

5.25.1 Detailed Description

Todo throw exceptions.

5.26 include/BCR_CPP_LA/class_extensions.decl File Reference

Declarations of general expansions for LA.

#include <stdexcept>
#include <sstream>
#include <iostream>
#include <linear_algebra.decl>

Namespaces

- namespace linear_algebra

Functions

- template<class TN> matrix< TN > & linear_algebra::zero (matrix< TN > &r)
- template<class TN> mat_full< TN > & linear_algebra::zero (mat_full< TN > &r)
- template<class TN> mat_sym_full< TN > & linear_algebra::zero (mat_sym_full< TN > &r)
- template<class TN> mat_asym_full< TN > & linear_algebra::zero (mat_asym_full< TN > &r)
- template<class TN> refvector< TN > & linear_algebra::zero (refvector< TN > &r)
- double & linear_algebra::zero (double &r)
- double & linear_algebra::one (double &r)
- template<class TN> matrix< TN > & linear_algebra::one (matrix< TN > &r)
- void linear_algebra::copy (double &a, const double &b, const long i)
- void linear_algebra::copy (double &a, const double &b)
- void linear_algebra::copy (long &a, const long &b, const long i)
- void linear_algebra::copy (long &a, const long &b)
- template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > &a, mat_sym_sparse< TN > &b, const long i)
- template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > &a, const mat_sym_sparse< TN > &b)
- template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > &a, mat_sym_sparse< TN > &b)
- template<class TN> void linear_algebra::copy (mat_sym_full< TN > &a, mat_sym_full< TN > &b, const long i)
- template<class TN> void linear_algebra::copy (mat_sym_full< TN > &a, const mat_sym_full< TN > &b)

- template<class TN> void linear_algebra::copy (mat_sym_full< TN > &a, mat_sym_full< TN > &b)
- template<class TN> void linear_algebra::copy (mat_full< TN > &a, mat_full< TN > &b, const long i)
- template<class TN> void linear_algebra::copy (mat_full< TN > &a, const mat_full< TN > &b)
- template<class TN> void linear_algebra::copy (mat_full< TN > &a, mat_full< TN > &b)
- template<class TN> void linear_algebra::copy (mat_sparse< TN > &a, mat_sparse< TN > &b, const long i)
- template<class TN> void linear_algebra::copy (mat_sparse< TN > &a, const mat_sparse< TN > &b)
- template<class TN> void linear_algebra::copy (mat_sparse< TN > &a, mat_sparse< TN > &b)
- template<class TN> void linear_algebra::copy (refvector< TN > &a, refvector< TN > &b, const long i)
- template<class TN> void linear_algebra::copy (refvector< TN > &a, const refvector< TN > &b)
- template<class TN> void linear_algebra::copy (refvector< TN > &a, refvector< TN > &b)
- template<class TN> void linear_algebra::copy (sparse_vector< TN > &a, sparse_vector< TN > &b, const long i)
- template<class TN> void linear_algebra::copy (sparse_vector< TN > &a, const sparse_vector< TN > &b)
- template<class C> void linear_algebra::copy (C &a, const C &b)
- template<class TN> void linear_algebra::display (const matrix< TN > &a, ostream &outs)
- template<class TN> void linear_algebra::display (const refvector< TN > &a, ostream &outs)
- void linear_algebra::display (const double &a, ostream &outs)

- template<class TNTN> mat_full< TNTN > linear_algebra::exp (const mat_asym_full< TNTN > &A)
- template<class TNTN> mat_full< TNTN > linear_algebra::dexp (mat_asym_full< TNTN > A, const long k, const long l)

5.26.1 Detailed Description

5.27 include/BCR_CPP_LA/class_extensions.hh File Reference

General expansions for LA.

#include <stdexcept>
#include <sstream>
#include <linear_algebra.decl>
#include <class_extensions.decl>
#include <refcount.decl>
#include <matrix.decl>
#include <mat_full.decl>
#include <mat_asym_full.decl>
#include <cmath>
#include <linear_algebra.h>

Namespaces

- namespace linear_algebra

Functions

- template<class TN> mat_sym_sparse< TN > & linear_algebra::zero (mat_sym_sparse< TN > &r)
- template<class TN> mat_sym_full< TN > & linear_algebra::zero (mat_sym_full< TN > &r)
- template<class TN> mat_sparse< TN > & linear_algebra::zero (mat_sparse< TN > &r)
- template<class TN> mat_full< TN > & linear_algebra::zero (mat_full< TN > &r)
- template<class TN> mat_asym_full< TN > & linear_algebra::zero (mat_asym_full< TN > &r)

- template<class TN> refvector< TN > & linear_algebra::zero (refvector< TN > &r)
- double & linear_algebra::zero (double &r)
- template<typename TN> TN & linear_algebra::zero (TN &r)
- double & linear_algebra::one (double &r)
- template<class TN> matrix< TN > & linear_algebra::one (matrix< TN > &r)
- void linear_algebra::copy (double &a, const double &b, const long i)
- void linear_algebra::copy (double &a, const double &b)
- void linear_algebra::copy (long &a, const long &b, const long i)
- void linear_algebra::copy (long &a, const long &b)
- void linear_algebra::copy (bool &a, const bool &b)
- template<class TN> void linear_algebra::copy (mat_full< TN > &a, const mat_full< TN > &b, const long i)
- template<class TN> void linear_algebra::copy (mat_full< TN > &a, const mat_full< TN > &b)
- template<class TN> void linear_algebra::copy (mat_full< TN > &a, mat_full< TN > &b)
- template<class TN> void linear_algebra::copy (mat_sparse< TN > &a, const mat_sparse< TN > &b, const long i)
- template<class TN> void linear_algebra::copy (mat_sparse< TN > &a, const mat_sparse< TN > &b)
- template<class TN> void linear_algebra::copy (mat_sparse< TN > &a, mat_sparse< TN > &b)
- template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > &a, const mat_sym_sparse< TN > &b, const long i)
- template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > &a, const mat_sym_sparse< TN > &b)
- template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > &a, mat_sym_sparse< TN > &b)

- template<class TN> void linear_algebra::copy (mat_sym_full< TN > &a, const mat_sym_full< TN > &b, const long i)
- template<class TN> void linear_algebra::copy (mat_sym_full< TN > &a, const mat_sym_full< TN > &b)
- template<class TN> void linear_algebra::copy (mat_sym_full< TN > &a, mat_sym_full< TN > &b)
- template<class TN> void linear_algebra::copy (refvector< TN > &a, refvector< TN > &b, const long i)
- template<class TN> void linear_algebra::copy (refvector< TN > &a, const refvector< TN > &b, const long i)
- template<class TN> void linear_algebra::copy (refvector< TN > &a, const refvector< TN > &b)
- template<class TN> void linear_algebra::copy (sparse_vector< TN > &a, sparse_vector< TN > &b, const long i)
- template<class TN> void linear_algebra::copy (sparse_vector< TN > &a, const sparse_vector< TN > &b)
- template<class C> void linear_algebra::copy (C &a, const C &b)
- template<class TN> void linear_algebra::display (const matrix< TN > &a, ostream &outs)
- template<class TN> void linear_algebra::display (const refvector< TN > &a, ostream &outs)
- void linear_algebra::display (const double &a, ostream &outs)
- template<class TNTN> mat_full< TNTN > linear_algebra::exp (const mat_asym_full< TNTN > &A)
- template<class TNTN> mat_full< TNTN > linear_algebra::dexp (mat_asym_full< TNTN > A, const long k, const long l)

5.27.1 Detailed Description

5.28 include/BCR_CPP_LA/invert_sym.h File Reference

Inversion of symmetric matrices.

#include <linear_algebra.h>
#include <vector>
#include <iostream>

- refvector< double > & invert_sym (const mat_sym_full< double > &S1, const refvector< double > &v, refvector< double > &r2)
  Invert a symmetric full matrix and multiply by a vector.
- refvector< double > invert_sym (const mat_sym_full< double > &S1, const refvector< double > &v)
  Invert a symmetric full matrix and multiply by a vector.
- mat_sym_full< double > invert_sym (mat_sym_full< double > &S)
  Invert a symmetric full matrix.

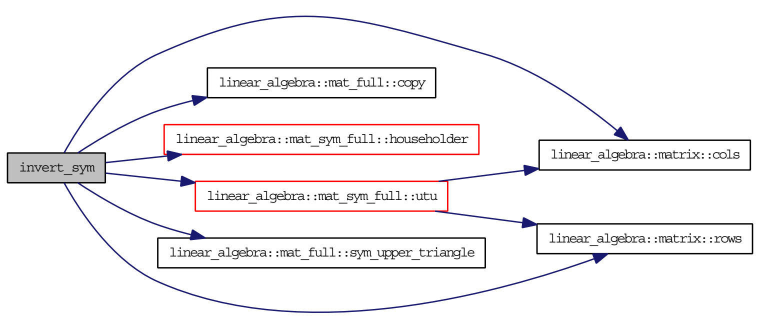
5.28.1 Detailed Description

5.29 Function Documentation

5.29.2.1 mat_sym_full<double> invert_sym (mat_sym_full< double > & S)[inline]

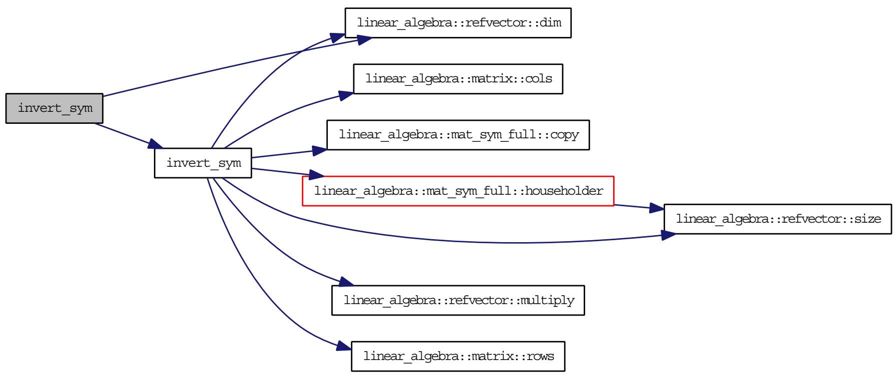
A symmetric matrix can be transformed to a tridiagonal matrix via a householder transformation ($O(n^2)$). A following gaussian elimination of the tridiagonal matrix (O(n)) can be used to solve for the inverted matrix.

Here is the call graph for this function:



5.29.2.2 refvector<double> invert_sym (const mat_sym_full< double > & *S1*, const refvector< double > & *v*)[inline]

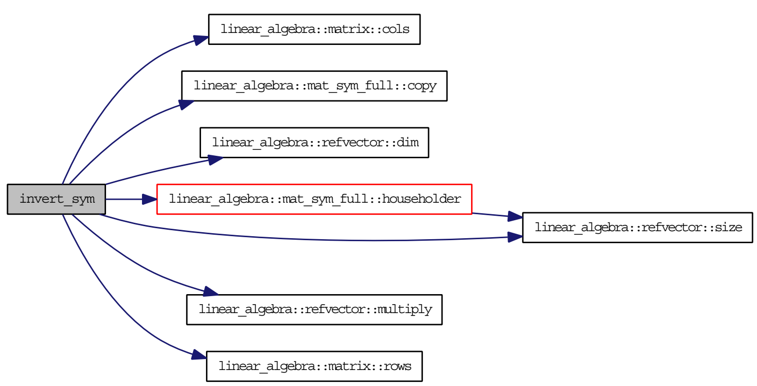
Here is the call graph for this function:



5.29.2.3 refvector<double>& invert_sym (const mat_sym_full< double > & *S1*, const refvector< double > & *v*, refvector< double > & *r2*)[inline]

A symmetric matrix can be transformed to a tridiagonal matrix via a householder transformation ($O(n^2)$). A following gaussian elimination of the tridiagonal matrix (O(n)). Can be used to solve for the inverted matrix.

Here is the call graph for this function:



5.30 include/BCR_CPP_LA/linear_algebra.decl File Reference

Namespace Class Declarations.

#include <refcount.decl>

#include <matrix.decl>
#include <square.decl>
#include <symmetric.decl>
#include <asymmetric.decl>
#include <sparse_vector.decl>
#include <mat_sparse.decl>
#include <mat_full.decl>
#include <mat_sym_full.decl>
#include <mat_sym_sparse.decl>
#include <mat_asym_sparse.decl>
#include <mat_asym_full.decl>
#include <class_extensions.decl>

Namespaces

- namespace linear_algebra

5.30.1 Detailed Description

5.31 include/BCR_CPP_LA/linear_algebra.h File Reference

#include <linear_algebra.decl>
#include <refcount.h>
#include <matrix.h>
#include <square.h>
#include <symmetric.h>
#include <asymmetric.h>
#include <sparse_vector.h>
#include <mat_sparse.h>
#include <mat_sym_full.h>
#include <mat_full.h>
#include <mat_sym_sparse.h>
#include <mat_asym_sparse.h>
#include <mat_asym_full.h>
#include <class_extensions.hh>

5.32 include/BCR_CPP_LA/mat_asym_full.decl File Reference

Provide declarations for antisymmetric full matrices.

#include <vector>
#include <linear_algebra.decl>
#include <asymmetric.decl>

Namespaces

- namespace linear_algebra

Classes

- class linear_algebra::mat_asym_full< TN >
  mat_asym_full uses packing for full antisymmetric matrices

5.32.1 Detailed Description

Todo
throw exceptions.

5.33 include/BCR_CPP_LA/mat_asym_full.h File Reference

Provide manipulations for antisymmetric full matrices.

#include <vector>
#include <linear_algebra.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <mat_full.decl>
#include <mat_sym_full.decl>
#include <mat_sym_sparse.decl>
#include <mat_asym_full.decl>
#include <cmath>
#include <refcount.h>
#include <sparse_vector.h>
#include <mat_full.h>
#include <mat_sym_full.h>
#include <mat_sym_sparse.h>

Namespaces

- namespace linear_algebra

5.33.1 Detailed Description

Todo

throw exceptions.

5.34 include/BCR_CPP_LA/mat_asym_sparse.decl File Reference

#include <vector>
#include <linear_algebra.decl>
#include <asymmetric.decl>

Namespaces

- namespace linear_algebra

Classes

- class linear_algebra::mat_asym_sparse< TN >
  mat_asym_sparse packs sparse column matrices into a comfortable format.

5.35 include/BCR_CPP_LA/mat_asym_sparse.h File Reference

#include <vector>
#include <linear_algebra.decl>
#include <mat_asym_sparse.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <mat_full.decl>
#include <mat_sparse.decl>
#include <mat_sym_full.decl>
#include <mat_asym_full.decl>
#include <refcount.h>
#include <sparse_vector.h>
#include <mat_full.h>
#include <mat_sparse.h>
#include <mat_sym_full.h>
#include <mat_asym_full.h>

Namespaces

- namespace linear_algebra

5.36 include/BCR_CPP_LA/mat_full.decl File Reference

Provide declarations for full matrix and vector interactions.

#include <vector>
#include <iostream>
#include <stdexcept>
#include <linear_algebra.decl>
#include <matrix.decl>

Namespaces

- namespace linear_algebra

Classes

- class linear_algebra::mat_full< TN >
  Class of full column matrices.

5.36.1 Detailed Description

5.37 include/BCR_CPP_LA/mat_full.h File Reference

#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <matrix.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <mat_sym_full.decl>
#include <mat_asym_full.decl>
#include <mat_sym_sparse.decl>
#include <refcount.h>
#include <matrix.h>
#include <sparse_vector.h>
#include <mat_sym_full.h>
#include <mat_asym_full.h>
#include <mat_sym_sparse.h>

Namespaces

- namespace linear_algebra

5.38 mat_full.h File Reference

#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <matrix.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <mat_sym_full.decl>
#include <mat_asym_full.decl>
#include <mat_sym_sparse.decl>
#include <refcount.h>
#include <matrix.h>
#include <sparse_vector.h>
#include <mat_sym_full.h>
#include <mat_asym_full.h>
#include <mat_sym_sparse.h>

Namespaces

- namespace linear_algebra

5.39 include/BCR_CPP_LA/mat_sparse.decl File Reference

Provide declarations for sparse matrix and vector interactions.

#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <matrix.decl>
#include <refcount.decl>

Namespaces

- namespace linear_algebra

Classes

- class linear_algebra::mat_sparse< TN >
  Class of sparse column matrices.

5.39.1 Detailed Description

5.40 include/BCR_CPP_LA/mat_sparse.h File Reference

Provide functionality for sparse matrix and vector interactions.

#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <matrix.decl>
#include <mat_sparse.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <mat_full.decl>
#include <mat_sym_sparse.decl>
#include <refcount.h>
#include <sparse_vector.h>
#include <mat_full.h>
#include <mat_sym_sparse.h>

Namespaces

- namespace linear_algebra

5.40.1 Detailed Description

5.41 include/BCR_CPP_LA/mat_sym_full.decl File Reference

Provide declarations for symmetric full matrices.

#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <symmetric.decl>

Namespaces

- namespace linear_algebra

Classes

- class linear_algebra::mat_sym_full< TN >
  mat_sym_full uses packing for full symmetric matrices

5.41.1 Detailed Description

Todo

throw exceptions.

5.42 include/BCR_CPP_LA/mat_sym_full.h File Reference

Provide manipulatins for symmetric full matrices.

#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <mat_full.decl>
#include <mat_asym_full.decl>
#include <mat_sym_full.decl>
#include <refcount.h>
#include <sparse_vector.h>
#include <mat_full.h>
#include <mat_asym_full.h>

Namespaces

- namespace linear_algebra

5.42.1 Detailed Description

Todo

throw exceptions.

5.43 include/BCR_CPP_LA/mat_sym_sparse.decl File Reference

Provide declarations for antisymmetric sparse matrices.

#include <vector>
#include <linear_algebra.decl>
#include <symmetric.decl>
#include "mat_sym_sparse_eigen.decl"

Namespaces

- namespace linear_algebra

Classes

- class linear_algebra::mat_sym_sparse< TN >
  mat_sym_sparse packs sparse column matrices into a comfortable format.

5.43.1 Detailed Description

5.44 include/BCR_CPP_LA/mat_sym_sparse.h File Reference

Provide functionality for symmetric sparse matrices.

#include <vector>
#include <linear_algebra.decl>
#include <mat_sym_sparse.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <mat_full.decl>
#include <mat_sym_full.decl>
#include <mat_asym_full.decl>
#include "mat_sym_sparse_eigen.h"
#include <refcount.h>
#include <sparse_vector.h>
#include <mat_full.h>
#include <mat_sym_full.h>
#include <mat_asym_full.h>

Namespaces

- namespace linear_algebra

5.44.1 Detailed Description

5.45 include/BCR_CPP_LA/mat_sym_sparse_eigen.decl File Reference

Eigenvalue related declarations.

Functions

- void gen_mat (const sparse_vector< TN > &a, const sparse_vector< TN > &b, TN correction)
  Produce the necessary $ij^* + ji^*$.
- mat_sparse< TN > householder (vector< TN > &diagonal, vector< TN > &subdiagonal)
  This routine produces a the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.
- mat_sparse< TN > householder (refvector< TN > &diagonal, refvector< TN > &subdiagonal)
  This routine produces a the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse ymmetric matrix.

5.45.1 Detailed Description

Todo

throw exceptions.

5.46 include/BCR_CPP_LA/mat_sym_sparse_eigen.h File Reference

Eigenvalue related routines.

5.46.1 Detailed Description

Todo

throw exceptions.

5.47 include/BCR_CPP_LA/matrix.decl File Reference

Declares fundamentals of matrices.

#include <vector>
#include <iostream>
#include <linear_algebra.decl>

Namespaces

- namespace linear_algebra

Classes

- class linear_algebra::matrix< TN >
  matrix is a basic class which covers all kinds of matrices.

Functions

- template<class TN> TN linear_algebra::operator * (const std::vector< TN > &a, const std::vector< TN > &b)
  Scalar product of two full vectors.

5.47.1 Detailed Description

5.48 include/BCR_CPP_LA/matrix.h File Reference

Describes fundamentals of matrices.

#include <vector>
#include <iostream>
#include <linear_algebra.decl>
#include <matrix.decl>

Namespaces

- namespace linear_algebra

Functions

- template<class TN> TN linear_algebra::operator * (const vector< TN > &a, const vector< TN > &b)
  Scalar product of two full vectors.

5.48.1 Detailed Description

5.49 include/BCR_CPP_LA/polynomial.decl File Reference

Class declarations for polynomial.

#include <linear_algebra.decl>
#include <sparse_vector_infty.decl>

Namespaces

- namespace linear_algebra

Classes

- class linear_algebra::polynomial< C >
  A class for polynomials.

5.49.1 Detailed Description

5.50 include/BCR_CPP_LA/polynomial.h File Reference

Impltextitentations for a class of polynomials.

#include <sparse_vector_infty.h>
#include <polynomial.decl>
#include <linear_algebra.h>
#include <class_extensions.h>

Namespaces

- namespace linear_algebra

5.50.1 Detailed Description

5.51 include/BCR_CPP_LA/refcount.decl File Reference

Contains the declarations to a reference counted vector class as well as general reference counting.

#include <vector>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include <linear_algebra.decl>

Namespaces

- namespace linear_algebra

Classes

- class linear_algebra::refcount< Type >
  Class refcount provides reference counted pointers. Most basic use.
- class linear_algebra::refvector< TN >
  Class refvector provides reference counted vectors.

Functions

- template<class TN> void std::operator>>(const vector< TN > &a, ostream &OUT)

5.51.1 Detailed Description

5.52 include/BCR_CPP_LA/refcount.h File Reference

Contains a reference counted vector class as well as general reference counting.

#include <vector>
#include <stdexcept>
#include <sstream>
#include <iostream>

#include <linear_algebra.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <class_extensions.hh>
#include <cstdlib>
#include <sparse_vector.h>

Namespaces

- namespace linear_algebra

Functions

- template<class TN> void std::operator>> (const vector< TN > &a, ostream &OUT)
- template<typename TN> void std::operator>> (const TN &a, ostream &OUT)

5.52.1 Detailed Description

5.53 include/BCR_CPP_LA/sparse_vector.decl File Reference

Provide declarations for sparse vector interactions.

#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <refcount.decl> %

#include <matrix.decl>

Namespaces

- namespace linear_algebra

Classes

- class linear_algebra::sparse_vector< TN >
  A class for sparse vectors.

Functions

- template<class TN> TN linear_algebra::operator * (sparse_vector< TN > a, std::vector< TN > b)
  Scalar product of a sparse_vector with a full vector.
- template<class TN> TN linear_algebra::operator * (sparse_vector< TN > a, refvector< TN > b)
  Scalar product of a sparse_vector with a full refvector.

5.53.1 Detailed Description

5.54 include/BCR_CPP_LA/sparse_vector.h File Reference

Provide functionality for sparse vectors.

#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <sparse_ector.decl>
#include <refcount.decl>
#include <cstdlib>
#include <refcount.h>

Namespaces

- namespace linear_algebra

Functions

- template<class TN> TN linear_algebra::operator * (sparse_vector< TN > a, vector< TN > b)
  Scalar product of a sparse_vector with a full vector.
- template<class TN> TN linear_algebra::operator * (sparse_vector< TN > a, refvector< TN > b)
  Scalar product of a sparse_vector with a full refvector.

5.54.1 Detailed Description

5.55 include/BCR_CPP_LA/sparse_vector_infty.decl File Reference

Provide declarations for sparse vector interactions of infinite dimensions.

#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <refcount.decl>

Namespaces

- namespace linear_algebra

Classes

- class linear_algebra::sparse_vector_infty< TN >
  A class for sparse vectors.

5.55.1 Detailed Description

5.56 include/BCR_CPP_LA/sparse_vector_infty.h File Reference

Provide functionality for sparse vectors of infinite dimension.

#include <vector>
#include <stdexcept>
#include <sparse_vector_infty.decl>
#include <refcount.h>
#include <class_extensions.h>

Namespaces

- namespace linear_algebra

5.56.1 Detailed Description

5.57 include/BCR_CPP_LA/square.decl File Reference

Declares fundamentals of square matrices.

#include <linear_algebra.decl>
#include <matrix.decl>

Namespaces

- namespace linear_algebra

Classes

- class linear_algebra::square< TN >
  The square matrix class contains all n by n matrices.

5.57.1 Detailed Description

5.58 include/BCR_CPP_LA/square.h File Reference

Describes fundamentals of matrices.

#include <vector>
#include <iostream>
#include <square.decl>
#include <matrix.h>

Namespaces

- namespace linear_algebra

5.58.1 Detailed Description

5.59 include/BCR_CPP_LA/symmetric.decl File Reference

Declares fundamentals of matrices.

#include <linear_algebra.decl>
#include <square.decl>

Namespaces

- namespace linear_algebra

Classes

- class linear_algebra::symmetric< TN >
  The symmetric class contains all symmetric matrices.

5.59.1 Detailed Description

5.60 include/BCR_CPP_LA/symmetric.h File Reference

Describes fundamentals of symmetric matrices.

#include <vector>
#include <iostream>
#include <square.h>
#include <symmetric.decl>

Namespaces

- namespace linear_algebra

5.60.1 Detailed Description

5.61 include/BCR_CPP_LA/trace_AB.hh File Reference

get the trace of matrix products.

#include <linear_algebra.h>

Namespaces

- namespace linear_algebra

Functions

- template<class TN> TN linear_algebra::trace_AB (const mat_sym_full< TN > &A, const mat_sym_full< TN > &B)
- template<class TN> TN linear_algebra::trace_AB (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B)
- template<class T> T linear_algebra::trace_AB (const mat_asym_full< T > &A, const mat_asym_full< T > &B)

5.61.1 Detailed Description

5.62 include/sorting_functions.hh File Reference

#include <BCR_CPP_LA/refcount.h>
#include <sorting_functions.hh>

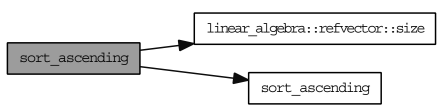
Functions

- template<class T1> static refvector< unsigned long > & sort_ascending (const refvector< T1 > &E, unsigned long start, unsigned long end, refvector< unsigned long > &index)
  Do a qsort on E with immediate storage back into index. ensure index has the right dimension. (perfectly parallelizable).
- template<class T1> static refvector< unsigned long > sort_ascending (const refvector< T1 > &E)

5.62.1 Function Documentation

5.62.1.1 template<class T1> static refvector<unsigned long> sort_ascending (const refvector< T1 > & *E*)[static]

Here is the call graph for this function:



5.62.1.2 template<class T1> static refvector<unsigned long>& sort_ascending (const refvector< T1 > & *E*, unsigned long *start*, unsigned long *end*, refvector< unsigned long > & *index*)[static]

5.63 include/stateful_thread_group.hh File Reference

#include <boost/thread/exceptions.hpp>
#include <ostream>

#include <boost/thread/detail/move.hpp>
#include <boost/thread/mutex.hpp>

#include <boost/thread/xtime.hpp>
#include <boost/thread/detail/thread_heap_alloc.hpp>
#include <boost/utility.hpp>
#include <boost/assert.hpp>
#include <list>
#include <algorithm>
#include <boost/ref.hpp>
#include <boost/cstdint.hpp>
#include <boost/bind.hpp>
#include <stdlib.h>
#include <memory>
#include <boost/utility/enable_if.hpp>
#include <boost/type_traits/remove_reference.hpp>
#include <boost/thread.hpp>
#include <boost/config/abi_prefix.hpp>
#include <boost/config/abi_suffix.hpp>

Namespaces

- namespace boost

Classes

- class boost::stateful_thread_group

5.64 stateful_thread_group.hh File Reference

#include <boost/thread/exceptions.hpp>
#include <ostream>
#include <boost/thread/detail/move.hpp>
#include <boost/thread/mutex.hpp>
#include <boost/thread/xtime.hpp>
#include <boost/thread/detail/thread_heap_alloc.hpp>
#include <boost/utility.hpp>
#include <boost/assert.hpp>
#include <list>
#include <algorithm>
#include <boost/ref.hpp>
#include <boost/cstdint.hpp>

#include <boost/bind.hpp>
#include <stdlib.h>
#include <memory>
#include <boost/utility/enable_if.hpp>
#include <boost/type_traits/remove_reference.hpp>
#include <boost/thread.hpp>
#include <boost/config/abi_prefix.hpp>
#include <boost/config/abi_suffix.hpp>

Namespaces

- namespace boost

Classes

- class boost::stateful_thread_group

5.65 isthreadableabstract.h File Reference

implementation of is_threadable_abstract

#include <boost/thread/shared_mutex.hpp>
#include <stateful_thread_group.hh>

Classes

- class is_threadable_abstract
  Abstract class implementing the Concepts::is_threadable concept.

5.65.1 Detailed Description

5.66 Library_data.cc File Reference

Implementations of the Library class for discrete spaces.

#include <BCR_CPP_LA/linear_algebra.h>
#include <Library_data.hh>
#include <boost/thread.hpp>
#include <boost/mem_fn.hpp>

5.66.1 Detailed Description

5.67 Library_data.hh File Reference

#include <BCR_CPP_LA/refcount.decl>
#include <typedefs.hh>
#include <string>

Classes

- class Library_data
  Abstract class for discrete optimizations.

5.68 noprune.h File Reference

Header of class noprune.

#include <concepts.hh>
#include <prunerabstract.h>

Classes

- class noprune< X >
  Dummy pruning class. No actual pruning done.

5.68.1 Detailed Description

5.69 optimizeabstract.h File Reference

implement abstract class optimize_abstract

Classes

- class optimize_abstract
  Abstract class that implements the Concepts::has_optimize concept. Only good for generic referncing of optimizations.

5.69.1 Detailed Description

5.70 prunerabstract.h File Reference

pruner_abstract implementation

#include <BCR_CPP_LA/refcount.h>
#include <concepts.hh>

Classes

- class pruner_abstract< X >
  Abstract class implementing the Concepts::pruner concept.

5.70.1 Detailed Description

5.71 reordergeneralbase.hh File Reference

#include <prunerabstract.h>
#include <typedefs.hh>
#include <BCR_CPP_LA/refcount.h>
#include <sorting_functions.hh>

Classes

- class reorder_general_base< X >

5.72 simpleprune.cc File Reference

Implementation file of simple_prune.

#include <BCR_CPP_LA/refcount.h>
#include <simpleprune.h>
#include <sorting_functions.hh>

5.72.1 Detailed Description

5.73 simpleprune.h File Reference

simple pruning class.

#include <typedefs.hh>
#include <boost/concept_check.hpp>
#include <concepts.hh>
#include <BCR_CPP_LA/refcount.decl>
#include <prunerabstract.h>

Classes

- class simple_prune< X >
  Class to prune a Library.

5.73.1 Detailed Description

5.74 typedefs.hh File Reference

#include <cmath>
#include <limits>

Classes

- struct valerg
  Pair of property value and energy.
- struct double_index
  Pair of configurational and conformational index, respectively.

Typedefs

- typedef unsigned long ulong

Functions

- bool operator== (const double_index &a, const double_index &b)
  Comparison operator.
- bool operator== (const valerg &a, const valerg &b)
  Comparison operator. item valerg operator- (const valerg &a, const valerg &b)
  Subtract two valerg.

- valerg operator-= (valerg &a, const valerg &b)
  Subtract two valerg.

- valerg operator+= (valerg &a, const valerg &b)
  Subtract two valerg.

- valerg operator+ (const valerg &a, const valerg &b)
  Add two valerg.

- valerg operator * (const valerg &a, const double b)
  Multiply.

- valerg & operator *= (valerg &r, const double b
) *Multiply.*

5.74.1 Typedef Documentation

5.74.1.1 typedef unsigned long ulong

5.74.2 Function Documentation

5.74.2.1 valerg operator * (const valerg & a, const double b)[inline]

5.74.2.2 valerg& operator *= (valerg & r, const double b)[inline]

5.74.2.3 valerg operator+ (const valerg & a, const valerg & b)[inline]

5.74.2.4 valerg operator+= (valerg & a, const valerg & b)[inline]

5.74.2.5 valerg operator- (const valerg & a, const valerg & b)[inline]

7.74.2.6 valerg operator-= (valerg & a, const valerg & b[inline]

5.74.2.7 bool operator== (const valerg & a, const valerg & b)[inline]

5.74.2.8 bool operator== (const double_index & a, const double_index & b)[inline]

Define zmat and zmat_entry.

#include <sstream>
#include <BCR_CPP_LA/linear_algebra.h>
#include <zmat.hh>

5.74.3 Detailed Description

5.75 zmat.hh File Reference

Define zmat and zmat_entry.

#include <sstream>
#include <BCR_CPP_LA/linear_algebra.h>

Classes

- class zmat_entry
  Class of Z-matrix entries.
- class zmat_connector
  Class of connectors between z-matrices.
- class zmat
  Class for Z-matrices. Just a wrapper class around a vector of entries.

5.75.1 Detailed Description

5.76 zmat_opt.cc File Reference

Z-matrix optimization class implementation.

#include <typedefs.hh>
#include <zmat.hh>
#include <zmat_opt.hh>
#include <iostream>
#include <fstream>

Functions

- valerg calc_property (const zmat &A, const string &out, const string &id, zmat &returnA)
  Setup the external computations and execute them.
- static valerg calc_energy (const zmat &A, const string &out, const string &id, zmat &returnA)
  Setup the external computations for the energy and execute them.

5.76.1 Detailed Description

5.76.2 Function Documentation

5.76.2.1 static valerg calc_energy (const zmat & *A*, const string & *out*, const string & *id*, zmat & *returnA*)[static]

requires

```
./energy_script
```

5.76.2.2 valerg calc_property (const zmat & *A*, const string & *out*, const string & *id*, zmat & *eturnA*)

5.77 zmat_opt.hh File Reference

Z-matrix optimization class definition.

#include <typedefs.hh>
#include <zmat.hh>
#include <Library_data.hh>
#include <noprune.h>
#include <isthreadableabstract.h>

Classes

- class zmat_opt

5.77.1 Detailed Description

6. Discrete Optimization in Chemical Space Example Documentation

6.1 carbazoles.inp

```
ChemGroup(
  (
   Z(
```

```
 (c, -3, 0.7, -2,   0, -1, 0)
 (c,  0, 1.4, -3, 126, -2, 0)
 (c,  1, 1.4,  0, 108, -3, 180)
 (c,  2, 1.4,  1, 108,  0, 0)
)
ReturnConnector()
 Connector(
  (
   (0,1,2)
   (0.7,0,0)
   (0,120,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
  (
   (-3,-2,-1)
   (0,0,0)
   (0,0,0)
   (0,0,0)

   (0,0,0)
   (0,0,0)
   (0,0,0)
   ()
  )
  (
   (-3,-2,-1)
   (0,0,0)
   (0,0,0)
   (0,0,0)

   (0,0,0)
   (0,0,0)
   (0,0,0)
   ()
  )
```

```
(
 (-3,-2,-1)
 (0,0,0)
 (0,0,0)
 (0,0,0)

 (0,0,0)
 (0,0,0)
 (0,0,0)
 ()
)
(
 (-3,-2,-1)
 (0,0,0)
 (0,0,0)
 (0,0,0)

 (0,0,0)
 (0,0,0)
 (0,0,0)
 ()
)
(
 (3,2,1)
 (0.7,0,0)
 (0,120,0)
 (0,0,180)

 (0,0,1)
 (0,0,1)
 (0,0,0)
 ()
)
(
 (-3,-2,-1)
 (0,0,0)
 (0,0,0)
 (0,0,0)

 (0,0,0)
```

```
   (0,0,0)
   (0,0,0)
   ()
  )
  (
   (-3,-2,-1)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   ()
  )
  (
   (-3,-2,-1)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   ()
  )
  (
   (-3,-2,-1)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   ()
  )
  (
   (0,1,2)
   (1.4,0,0)
```

```
    (0, 108, 0)
    (0,   0, 0)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
   )
  )
  allowed_groups(
  (7,2,3,4,5,6,8,9,10,11,12)
  (7,2,3,4,5,6,8,9,10,11,12)
  (7,2,3,4,5,6,8,9,10,11,12)
  (7,2,3,4,5,6,8,9,10,11,12)
  (1)
  (7,2,3,4,5,6,8,9,10,11,12)
  (7,2,3,4,5,6,8,9,10,11,12)
  (7,2,3,4,5,6,8,9,10,11,12)
  (7,2,3,4,5,6,8,9,10,11,12)
  (1)
  (13,14,15)
  )
 )
 (
  Z()
  ReturnConnector(
   (
    (-6,-5,-4)
    (0,0, 0)
    (0,0, 0)
    (0,0, 0)

    (0,0,0)
    (0,0,0)
    (0,0,0)
    ()
   )
  )
  Connector()
  allowed_groups()
```

```
)
(
 Z(
 (C, -3, 0.7, -2,   0, -1, 0)
 (C,  0, 1.4, -3, 120, -2, 180)
 (N,  0, 2.7, -3, 120, -2, 180)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7, 0, 0)
   (0, 120, 180)
   (0,0, 180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
 )
 Connector()
 allowed_groups()
)
(
 Z(
 (C, -3, 0.7, -2,   0, -1,   0)
 (C,  0, 1.4, -3, 120, -2, 180)
 (O,  1, 1.4,  0, 120, -3, -91)
 (H,  1, 1.1,  2, 120,  0, 180)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7, 0, 0)
   (0, 120, 180)
   (0,0, 180)

   (0,0,1)
   (0,0,0)
   (0,0,0)
```

```
    ()
   )
  )
  Connector()
  allowed_groups()
 )
 (
  Z(
  (C, -3, 0.7, -2,   0,    -1,   0)
  (O,  0, 1.4, -3, 120, -2, 180)
  (C,  1, 1.4,  0, 120,    -3, -91(180))
  (H,  2, 1.1,  1, 109.47,  0, 180)
  (H,  2, 1.1,  1, 109.47,  3, 120)
  (H,  2, 1.1,  1, 109.47,  4, 120)
  )
  ReturnConnector(
   (
    (0,1,-3)
    (0.7, 0, 0)
    (0, 120, 180)
    (0,0, 180)

    (0,0,1)
    (0,0,0)
    (0,0,0)
    ()
   )
  )
  Connector()
  allowed_groups()
 )
 (
  Z(
  (C, -3, 0.7, -2,   0, -1,   0)
  (C,  0, 1.4, -3, 120, -2, 180)
  (O,  1, 1.3,  0, 120, -3, -91(180))
  (C,  1, 1.5,  2, 120,  0, 180)
  (H,  3, 1.1,  1, 109.47,  2, 180)
  (H,  3, 1.1,  1, 109.47,  4, 120)
  (H,  3, 1.1,  1, 109.47,  5, 120)
```

```
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7, 0, 0)
   (0, 120, 180)
   (0,0, 180)

   (0,0,1)
   (0,0,0)
   (0,0,0)
   ()
  )
 )
 Connector()
 allowed_groups()
)
(
 Z(
 (C, -3, 0.7, -2,   0,    -1,   0)
 (N,  0, 1.4, -3, 120,    -2, 180)
 (C,  1, 1.4,  0, 120,    -3, -91(180))
 (O,  2, 1.3,  1, 120,     0,   0(180))
 (C,  2, 1.5,  1, 120,     3, 180)
 (H,  4, 1.1,  2, 109.47,  1, 180)
 (H,  4, 1.1,  2, 109.47,  5, 120)
 (H,  4, 1.1,  2, 109.47,  6, 120)

 (H,  1, 1.1, 2, 120,  0, 180)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7, 0, 0)
   (0, 120, 180)
   (0,0, 180)

   (0,0,1)
   (0,0,0)
   (0,0,0)
```

```
   ()
  )
 )
 Connector()
 allowed_groups()
)
(
 Z(
 (C,  -3, 0.7, -2, 0,    -1, 0)
 (H,   0, 1.1, -3, 120, -2, 180)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7, 0, 0)
   (0, 120, 180)
   (0,0, 180)

   (0,0,1)
   (0,0,0)
   (0,0,0)
   ()
  )
 )
 Connector()
 allowed_groups()
)
(
 Z(
 (C, -3, 0.7, -2,   0, -1,   0)
 (N,  0, 1.4, -3, 120,    -2, 180)
 (O,  1, 1.2,  0, 120, -3, -80)
 (O,  1, 1.2,  2, 120,  0, 180)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7, 0, 0)
   (0, 120, 180)
   (0,0, 180)
```

```
   (0,0,1)
   (0,0,0)
   (0,0,0)
   ()
  )
 )
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2,   0, -1,   0)
 (O,  0, 1.3, -3, 120,    -2, 180)
  (c,  1, 1.4,  0, 120, -3,   0(180))
  (o,  2, 1.3,  1, 120,  0,   0(180))
  (c,  2, 1.5,  1, 120,  3, 180)
  (H,  4, 1.1,  2, 109.47, 1, 180)
  (h,  4, 1.1,  2, 109.47, 5, 120)
  (h,  4, 1.1,  2, 109.47, 6, 120)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7, 0, 0)
   (0, 120, 180)
   (0,0, 180)

   (0,0,1)
   (0,0,0)
   (0,0,0)
   ()
  )
 )
 Connector()
 allowed_groups()
)
(
 Z(
 (C, -3, 0.7, -2,   0,    -1,   0)
```

```
  (O,  0, 1.4, -3, 120,    -2, 180)
  (H,  1, 1.1,  0, 120,    -3, -91(180))
  )
  ReturnConnector(
   (
    (0,1,-3)
    (0.7, 0, 0)
    (0, 120, 180)
    (0,0, 180)

    (0,0,1)
    (0,0,0)
    (0,0,0)
    ()
   )
  )
  Connector()
  allowed_groups()
 )
 (
  Z(
  (C, -3, 0.7, -2,   0   ,    -1,   0)
  (N,  0, 1.4, -3, 120,    -2, 180)
  (H,  1, 1.1,  0, 120   ,    -3, 0)
  (H,  1, 1.1,  2, 109.47,     0, 120(-240))
  )
  ReturnConnector(
   (
    (0,1,-3)
    (0.7, 0, 0)
    (0, 120, 180)
    (0,0, 180)

    (0,0,1)
    (0,0,0)
    (0,0,0)
    ()
   )
  )
  Connector()
```

```
 allowed_groups()
)
(
 Z(
 (C, -3, 0.7, -2,   0,   -1,   0)
 (N,  0, 1.4, -3, 120,    -2, 180)
 (C,  1, 1.4,  0, 120,   -3,  60)
 (C,  1, 1.4,  2, 120,    0, 120(-240))
 (H,  2, 1.1,  1, 109.47, 3, 180)
 (H,  2, 1.1,  1, 109.47, 4, 120)
 (H,  2, 1.1,  1, 109.47, 5, 120)
 (H,  3, 1.1,  1, 109.47, 2,  80)
 (H,  3, 1.1,  1, 109.47, 7, 120)
 (H,  3, 1.1,  1, 109.47, 8, 120)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7, 0, 0)
   (0, 120, 180)
   (0,0, 180)

   (0,0,1)
   (0,0,0)
   (0,0,0)
   ()
  )
 )
 Connector()
 allowed_groups()
)
(
      Z(
      (o, -3, 0, -2,0,-1,0)
      )
      ReturnConnector()
      Connector()
      allowed_groups()
)
(
```

```
        Z(
        (s, -3, 0, -2,0,-1,0)
        )
        ReturnConnector()
        Connector()
        allowed_groups()
  )
  (
        Z(
        (c, -3, 0, -2, 0,-1,0)
        (o,  0, 1.3, -3, 126, -2, 180)
        )
        ReturnConnector()
        Connector()
        allowed_groups()
  )
)
```

6.2 ChemGroup

Default construction. ChemIdent(”H”)

6.3 vanilla-rings.inp

```
ChemGroup(
  (
   Z()
   ReturnConnector()
   Connector(
    (
     (-3,-2,-1)
     (0,0,0)
     (0,0,0)
     (0,0,0)

     (0,0,0)
     (0,0,0)
     (0,0,0)
     ()
    )
```

```
(
 (-3,-2,-1)
 (0,0,0)
 (0,0,0)
 (0,0,0)
 (0,0,0)
 (0,0,0)
 (0,0,0)
 ()
)
(
 (-3,-2,-1)
 (0,0,0)
 (0,0,0)
 (0,0,0)
 (0,0,0)
 (0,0,0)
 (0,0,0)
 ()
)
(
 (-3,-2,-1)
 (0,0,0)
 (0,0,0)
 (0,0,0)
 (0,0,0)
 (0,0,0)
 (0,0,0)
 ()
)
(
 (-3,-2,-1)
 (0,0,0)
 (0,0,0)
 (0,0,0)
 (0,0,0)
```

```
   (0,0,0)
   (0,0,0)
   ()
  )
  (
   (-3,-2,-1)
   (0,0,0)
   (0,0,0)
   (0,0,0)

   (0,0,0)
   (0,0,0)
   (0,0,0)
   ()
  )
  (
   (-3,-2,-1)
   (0,0,0)
   (0,0,0)
   (0,0,0)

   (0,0,0)
   (0,0,0)
   (0,0,0)
   ()
  )
 )
  (36,37,38,39,40,41,42,43,44,45,46)
  (47,1,2, 7,12)
  (47,1,3, 8,13)
  (47,1,4, 9,14)
  (47,1,5,10,15)
  (47,1,6,11,16)
  (37,36,38,39,40,41,42,43,44,45,46)
 )
)
(
 Z(
   (C, -3, 0.7, -2, 0, -1, 0 )
   (C,  0, 1.3, -3, 170.0, -2, 180 )
```

```
)
ReturnConnector(
  (
   (1,0,-3)
   (0.7, 0, 0)
   (  0, 179.0, 0)
   (  0, 0, 180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
 )
 Connector()
 allowed_groups()
)

(
 Z(
 )
 ReturnConnector(
  (
   (-3,-2,-1)
   (2.2,     0,   0)
   (  0, 30.0,   0)
   (  0,   0  , 0)

   (0,0,0)
   (0,0,0)
   (0,0,0)
   ()
  )
 )
 Connector(
  (
   (-3,-2,-1)
   (0, 0, 0)
   (0, 0, 0)
   (0, 0, 0)
```

```

    (0, 0, 1)
    (0, 0, 1)
    (0, 0, 1)
    ()
   )
   (
    (-3,-2,-1)
    (0.9, 0, 0)
    (0, 10, 180)
    (0, 0, 0)

    (0, 0, 1)
    (0, 0, 1)
    (0, 0, 1)
    ()
   )
  )
  allowed_groups(
   (48,49,50,51,52,53,54,55,56,57,58)
   (48,49,50,51,52,53,54,55,56,57,58)
  )
 )
 (
  Z(
  )
  ReturnConnector(
   (
    (-3,-2,-1)
    (2.2,     0,   0)
    (  0, 30.0,   0)
    (  0,   0  , 0)

    (0,0,0)
    (0,0,0)
    (0,0,0)
    ()
   )
  )
  Connector(
```

```
  (
   (-3,-2,-1)
   (0, 0, 0)
   (0, 0, 0)
   (0, 0, 0)

   (0, 0, 1)
   (0, 0, 1)
   (0, 0, 1)
   ()
  )
  (
   (-3,-2,-1)
   (0.9, 0, 0)
   (0, 10, 180)
   (0, 0, 0)

   (0, 0, 1)
   (0, 0, 1)
   (0, 0, 1)
   ()
  )
 )
 allowed_groups(
  (48,49,50,51,52,53,54,55,56,57,58)
  (48,49,50,51,52,53,54,55,56,57,58)
 )
)
(
 Z(
 )
 ReturnConnector(
  (
   (-3,-2,-1)
   (2.2,     0,   0)
   (  0, 30.0,   0)
   (  0,   0  , 0)

   (0,0,0)
   (0,0,0)
```

```
   (0,0,0)
   ()
  )
 )
 Connector(
  (
   (-3,-2,-1)
   (0, 0, 0)
   (0, 0, 0)
   (0, 0, 0)

   (0, 0, 1)
   (0, 0, 1)
   (0, 0, 1)
   ()
  )
  (
   (-3,-2,-1)
   (0.9, 0, 0)
   (0, 10, 180)
   (0, 0, 0)

   (0, 0, 1)
   (0, 0, 1)
   (0, 0, 1)
   ()
  )
 )
 allowed_groups(
  (48,49,50,51,52,53,54,55,56,57,58)
  (48,49,50,51,52,53,54,55,56,57,58)
 )
)
(
 Z(
 )
 ReturnConnector(
  (
   (-3,-2,-1)
   (2.2,    0,   0)
```

```
   (  0, 30.0,   0)
   (  0,   0  , 0)

   (0,0,0)
   (0,0,0)
   (0,0,0)
   ()
  )
 )
 Connector(
  (
   (-3,-2,-1)
   (0, 0, 0)
   (0, 0, 0)
   (0, 0, 0)

   (0, 0, 1)
   (0, 0, 1)
   (0, 0, 1)
   ()
  )
  (
   (-3,-2,-1)
   (0.9, 0, 0)
   (0, 10, 180)
   (0, 0, 0)

   (0, 0, 1)
   (0, 0, 1)
   (0, 0, 1)
   ()
  )
 )
 allowed_groups(
  (48,49,50,51,52,53,54,55,56,57,58)
  (48,49,50,51,52,53,54,55,56,57,58)
 )
)
(
 Z(
```

```
)
ReturnConnector(
 (
  (-3,-2,-1)
  (2.2,     0,   0)
  (  0, 30.0,   0)
  (  0,   0  , 0)

  (0,0,0)
  (0,0,0)
  (0,0,0)
  ()
 )
)
Connector(
 (
  (-3,-2,-1)
  (0, 0, 0)
  (0, 0, 0)
  (0, 0, 0)

  (0, 0, 1)
  (0, 0, 1)
  (0, 0, 1)
  ()
 )
 (
  (-3,-2,-1)
  (0.9, 0, 0)
  (0, 10, 180)
  (0, 0, 0)

  (0, 0, 1)
  (0, 0, 1)
  (0, 0, 1)
  ()
 )
)
allowed_groups(
 (48,49,50,51,52,53,54,55,56,57,58)
```

```
  (48,49,50,51,52,53,54,55,56,57,58)
 )
)

(
 Z(
   (C, -3, 0.7, -2,  0, -1,   0 )
   (C, -3, 3.7,  0,1.0, -1, 180 )
 )
 ReturnConnector(
  (
   (1,0,-3)
   (0.7, 0, 0)
   (  0, 178.0, 0)
   (  0, 0, 180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
 )
 Connector(
  (
   (0,1,-3)
   (0.7,0,0)
   (0,60,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
  (
   (0,1,-3)
   (0.7,0,0)
   (0,60,0)
   (0,0,0)

```

```
   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
  (
   (1,0,-3)
   (0.7,0,-180)
   (0,60,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
  (
   (1,0,-3)
   (0.7,0,-180)
   (0,60,)
   (0,0,0)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
)
 allowed_groups(
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
 )
)
(
 Z(
   (C, -3, 0.7, -2,  0, -1,   0 )
   (C, -3, 3.7,  0,1.0, -1, 180 )
 )
```

```
ReturnConnector(
 (
  (1,0,-3)
  (0.7, 0, 0)
  (  0, 178.0, 0)
  (  0, 0, 180)

  (0,0,1)
  (0,0,1)
  (0,0,0)
  ()
 )
)
Connector(
 (
  (0,1,-3)
  (0.7,0,0)
  (0,60,0)
  (0,0,180)

  (0,0,1)
  (0,0,1)
  (0,0,0)
  ()
 )
 (
  (0,1,-3)
  (0.7,0,0)
  (0,60,0)
  (0,0,0)

  (0,0,1)
  (0,0,1)
  (0,0,0)
  ()
 )
 (
  (1,0,-3)
  (0.7,0,-180)
  (0,60,0)
```

```
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
  (
   (1,0,-3)
   (0.7,0,-180)
   (0,60,)
   (0,0,0)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
)
 allowed_groups(
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
 )
)
(
 Z(
   (C, -3, 0.7, -2,  0, -1,   0 )
   (C, -3, 3.7,  0,1.0, -1, 180 )
 )
 ReturnConnector(
  (
   (1,0,-3)
   (0.7, 0, 0)
   (  0, 178.0, 0)
   (  0, 0, 180)

   (0,0,1)
   (0,0,1)
```

```
  (0,0,0)
  ()
 )
)
Connector(
 (
  (0,1,-3)
  (0.7,0,0)
  (0,60,0)
  (0,0,180)

  (0,0,1)
  (0,0,1)
  (0,0,0)
  ()
 )
 (
  (0,1,-3)
  (0.7,0,0)
  (0,60,0)
  (0,0,0)

  (0,0,1)
  (0,0,1)
  (0,0,0)
  ()
 )
 (
  (1,0,-3)
  (0.7,0,-180)
  (0,60,0)
  (0,0,180)

  (0,0,1)
  (0,0,1)
  (0,0,0)
  ()
 )
 (
  (1,0,-3)
```

```
   (0.7,0,-180)
   (0,60,)
   (0,0,0)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
)
 allowed_groups(
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
 )
)
(
 Z(
   (C, -3, 0.7, -2,  0, -1,   0 )
   (C, -3, 3.7,  0,1.0, -1, 180 )
 )
 ReturnConnector(
  (
   (1,0,-3)
   (0.7, 0, 0)
   (  0, 178.0, 0)
   (  0, 0, 180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
 )
 Connector(
  (
   (0,1,-3)
   (0.7,0,0)
   (0,60,0)
```

```
  (0,0,180)

  (0,0,1)
  (0,0,1)
  (0,0,0)
  ()
 )
 (
  (0,1,-3)
  (0.7,0,0)
  (0,60,0)
  (0,0,0)

  (0,0,1)
  (0,0,1)
  (0,0,0)
  ()
 )
 (
  (1,0,-3)
  (0.7,0,-180)
  (0,60,0)
  (0,0,180)

  (0,0,1)
  (0,0,1)
  (0,0,0)
  ()
 )
 (
  (1,0,-3)
  (0.7,0,-180)
  (0,60,)
  (0,0,0)

  (0,0,1)
  (0,0,1)
  (0,0,0)
  ()
 )
```

```
)
 allowed_groups(
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
 )
)
(
 Z(
   (C, -3, 0.7, -2,  0, -1,   0 )
   (C, -3, 3.7,  0,1.0, -1, 180 )
 )
 ReturnConnector(
  (
   (1,0,-3)
   (0.7, 0, 0)
   (  0, 178.0, 0)
   (  0, 0, 180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
 )
 Connector(
  (
   (0,1,-3)
   (0.7,0,0)
   (0,60,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
  (
   (0,1,-3)
```

```
   (0.7,0,0)
   (0,60,0)
   (0,0,0)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
  (
   (1,0,-3)
   (0.7,0,-180)
   (0,60,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
  (
   (1,0,-3)
   (0.7,0,-180)
   (0,60,)
   (0,0,0)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
)
 allowed_groups(
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
  (17,18,19,20,21,22,23,24,25,26,27)
 )
)
```

```
(
 Z(
  (C, -3, 0.7, -2, 0, -1,   0)
  (C,  0, 2.5, -3,162, -2, 180)
 )
  ReturnConnector(
   (
    (1,0,-3)
    (0.7, 0, 0)
    (  0, 150, 0)
    (  0, 0, 0)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
   )
  )
  Connector(
   (
    (1,0,-3)
    (0.7, 0, 0)
    (0, 36, 0)
    (0,0,0)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
   )
   (
    (0,1,-3)
    (0.7, 0, 0)
    (0, 72, 0)
    (0,0,180)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
```

```
   )
   (
    (1,0,-3)
    (0.7, 0, 0)
    (0, 72, 0)
    (0,0,180)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
   )
  )
  allowed_groups(
   (28,29,30,31,32,33,34,35)
   (17,18,19,20,21,22,23,24,25,26,27)
   (17,18,19,20,21,22,23,24,25,26,27)
  )
)
(
 Z(
  (C, -3, 0.7, -2, 0, -1,   0)
  (C,  0, 2.5, -3,162, -2, 180)
 )
  ReturnConnector(
   (
    (1,0,-3)
    (0.7, 0, 0)
    (  0, 150, 0)
    (  0, 0, 0)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
   )
  )
  Connector(
   (
    (1,0,-3)
```

```
    (0.7, 0, 0)
    (0, 36, 0)
    (0,0,0)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
   )
   (
    (0,1,-3)
    (0.7, 0, 0)
    (0, 72, 0)
    (0,0,180)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
   )
   (
    (1,0,-3)
    (0.7, 0, 0)
    (0, 72, 0)
    (0,0,180)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
   )
  )
  allowed_groups(
   (28,29,30,31,32,33,34,35)
   (17,18,19,20,21,22,23,24,25,26,27)
   (17,18,19,20,21,22,23,24,25,26,27)
  )
)
(
 Z(
```

```
 (C, -3, 0.7, -2, 0, -1,   0)
 (C,  0, 2.5, -3,162, -2, 180)
)
 ReturnConnector(
  (
   (1,0,-3)
   (0.7, 0, 0)
   (  0, 150, 0)
   (  0, 0, 0)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
 )
 Connector(
  (
   (1,0,-3)
   (0.7, 0, 0)
   (0, 36, 0)
   (0,0,0)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
  (
   (0,1,-3)
   (0.7, 0, 0)
   (0, 72, 0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
  (
```

```
    (1,0,-3)
    (0.7, 0, 0)
    (0, 72, 0)
    (0,0,180)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
   )
  )
  allowed_groups(
   (28,29,30,31,32,33,34,35)
   (17,18,19,20,21,22,23,24,25,26,27)
   (17,18,19,20,21,22,23,24,25,26,27)
  )
)
(
 Z(
  (C, -3, 0.7, -2, 0, -1,   0)
  (C,  0, 2.5, -3,162, -2, 180)
 )
  ReturnConnector(
   (
    (1,0,-3)
    (0.7, 0, 0)
    (  0, 150, 0)
    (  0, 0, 0)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
   )
  )
  Connector(
   (
    (1,0,-3)
    (0.7, 0, 0)
    (0, 36, 0)
```

```
    (0,0,0)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
   )
   (
    (0,1,-3)
    (0.7, 0, 0)
    (0, 72, 0)
    (0,0,180)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
   )
   (
    (1,0,-3)
    (0.7, 0, 0)
    (0, 72, 0)
    (0,0,180)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
   )
  )
  allowed_groups(
   (28,29,30,31,32,33,34,35)
   (17,18,19,20,21,22,23,24,25,26,27)
   (17,18,19,20,21,22,23,24,25,26,27)
  )
)
(
 Z(
  (C, -3, 0.7, -2, 0, -1,   0)
  (C,  0, 2.5, -3,162, -2, 180)
```

```
)
 ReturnConnector(
  (
   (1,0,-3)
   (0.7, 0, 0)
   (  0, 150, 0)
   (  0, 0, 0)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
 )
 Connector(
  (
   (1,0,-3)
   (0.7, 0, 0)
   (0, 36, 0)
   (0,0,0)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
  (
   (0,1,-3)
   (0.7, 0, 0)
   (0, 72, 0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
  (
   (1,0,-3)
   (0.7, 0, 0)
```

```
    (0, 72, 0)
    (0,0,180)

    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
   )
  )
  allowed_groups(
   (28,29,30,31,32,33,34,35)
   (17,18,19,20,21,22,23,24,25,26,27)
   (17,18,19,20,21,22,23,24,25,26,27)
  )
)

(
 Z(
  (C, -3, 0.7, -2, 0, -1, 0)
  (H,  0, 1.1, -3, 120, -2, 180)
 )
 ReturnConnector()
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2,   0, -1,   0)
  (N,  0, 1.4, -3, 120, -2, 180)
  (H,  1, 1.1,  0, 120, -3, 150)
  (H,  1, 1.1,  0, 120,  2, 150)
 )
 ReturnConnector()
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2,   0, -1,   0)
  (N,  0, 1.4, -3, 120, -2, 180)
```

```
  (O,  1, 1.2,  0, 120, -3, 150)
  (O,  1, 1.2,  0, 120,  2, 180)
 )
 ReturnConnector()
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2,   0, -1,   0)
  (N,  0, 1.4, -3, 120, -2, 180)
  (H,  1, 1.1,  0, 120, -3, 150)
  (C,  1, 1.4,  0, 120,  2, 180)
  (O,  3, 1.3,  1, 120,  2, 0(180))
  (C,  3, 1.5,  1, 120,  4, 180)
  (H,  5, 1.1,  3, 109.47, 4,180)
  (H,  5, 1.1,  3, 109.47, 6, 120)
  (H,  5, 1.1,  3, 109.47, 7, 120)
 )
 ReturnConnector()
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2,   0,   -1,   0)
  (C,  0, 1.4, -3, 120,   -2, 180)
  (O,  1, 1.3,  0, 120,   -3, -30(150))
  (C,  1, 1.5,  0, 120,    2, 180)
  (H,  3, 1.1,  1, 109.47, 2, 180)
  (H,  3, 1.1,  1, 109.47, 4, 120)
  (H,  3, 1.1,  1, 109.47, 5, 120)
 )
 ReturnConnector()
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2,   0,   -1,   0)
```

```
   (O,  0, 1.4, -3, 120,   -2, 180)
   (C,  1, 1.4,  0, 120,   -3, -30(150))
   (O,  2, 1.3,  1, 120,    0, 0(180))
   (C,  2, 1.5,  1, 120,    3, 180)
   (H,  4, 1.1,  2, 109.47, 3,180)
   (H,  4, 1.1,  2, 109.47, 5, 120)
   (H,  4, 1.1,  2, 109.47, 6, 120)
  )
  ReturnConnector()
  Connector()
  allowed_groups()
 )
 (
  Z(
   (C, -3, 0.7, -2, 0, -1, 0)
   (C,  0, 1.5, -3, 120, -2, 180)
   (N,  0, 2.8, -3, 170, -2, 180)
  )
  ReturnConnector()
  Connector()
  allowed_groups()
 )
 (
  Z(
   (C, -3, 0.7, -2,   0, -1,   0)
   (N,  0, 1.4, -3, 120, -2, 180)
   (C,  1, 1.2,  0, 120, -3, 150)
   (C,  1, 1.2,  0, 120,  2, 180)

   (H,  2, 1.1,  1, 109.47, 0,180)
   (H,  2, 1.1,  1, 109.47, 4, 120)
   (H,  2, 1.1,  1, 109.47, 5, 120)

   (H,  3, 1.1,  1, 109.47, 0,180)
   (H,  3, 1.1,  1, 109.47, 7, 120)
   (H,  3, 1.1,  1, 109.47, 8, 120)
  )
  ReturnConnector()
  Connector()
  allowed_groups()
```

```
)
(
 Z(
  (C, -3, 0.7, -2,   0, -1,   0)
  (O,  0, 1.4, -3, 120, -2, 180)
  (C,  1, 1.5,  0, 120, -3, -30(150))
  (H,  2, 1.1,  1, 109.47, 0,180)
  (H,  2, 1.1,  1, 109.47, 3, 120)
  (H,  2, 1.1,  1, 109.47, 4, 120)
 )
 ReturnConnector()
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2,   0, -1,   0)
  (S,  0, 1.6, -3, 120, -2, 180)
  (C,  1, 1.5,  0, 120, -3, -30(150))
  (H,  2, 1.1,  1, 109.47, 0,180)
  (H,  2, 1.1,  1, 109.47, 3, 120)
  (H,  2, 1.1,  1, 109.47, 4, 120)
 )
 ReturnConnector()
 Connector()
 allowed_groups()
)
(
 Z(
  (N, -3,0.6, -2,0,-1,0)
 )
 ReturnConnector()
 Connector()
 allowed_groups()
)


 (
  Z(
   (O, -3, 0.7, -2,   0   , -1,    0)
```

```
  )
  ReturnConnector()
  Connector()
  allowed_groups()
 )
 (
  Z(
   (S, -3, 0.7, -2,   0   , -1,    0)
  )
  ReturnConnector()
  Connector()
  allowed_groups()
 )
 (
  Z(
   (Si, -3,  0.7, -2,   0   , -1,    0)
   (H ,  0,1.4, -3, 109.47, -2,  120)
   (H ,  0,1.4,  1, 109.47, -3, -120)
  )
  ReturnConnector()
  Connector()
  allowed_groups()
 )
 (
  Z(
   (S, -3,  0.7, -2,   0   , -1,    0)
   (O ,  0,1.9, -3, 109.47, -2,  120)
   (O ,  0,1.9,  1, 109.47, -3, -120)
  )
  ReturnConnector()
  Connector()
  allowed_groups()
 )
 (
  Z(
   (C, -3,  0.7, -2,   0   , -1,    0)
   (H ,  0,1.1, -3, 109.47, -2,  120)
   (H ,  0,1.1,  1, 109.47, -3, -120)
  )
  ReturnConnector()
```

```
  Connector()
  allowed_groups()
 )
 (
  Z(
   (C, -3,  0.7, -2,   0   , -1,    0)
   (O , 0,1.3, -3, 120   , -2(1), -180)
  )
  ReturnConnector()
  Connector()
  allowed_groups()
 )
 (
  Z(
   (N, -3,  0.7, -2,   0   , -1,    0)
   (C ,  0,1.5, -3, 120.00, -2, -180)
   (H ,  1,1.1,  0, 109.47, -3,  60)
   (H ,  1,1.1,  0, 109.47,  2, 120)
   (H ,  1,1.1,  0, 109.47,  3, 120)
  )
  ReturnConnector()
  Connector()
  allowed_groups()
 )
 (
  Z(
   (C , -3,  0.7, -2,   0   , -1,    0)
   (C ,  0,1.4, -3, 120   , -2,  180)
   (H ,  1,1.1,  0, 120.0 , -3,  180)
   (H ,  1,1.1,  0, 120.0 ,  2,  180)
  )
  ReturnConnector()
  Connector()
  allowed_groups()
 )


 (
  Z(
  (N, -3, 0.7, -2,   0,   -1,   0)
```

```
 (C,  0, 1.4, -3, 120,   -2,  60)
 (C,  0, 1.4,  1, 120,   -3, 120(-240))
 (H,  1, 1.1,  0, 109.47, 2(1), 180)
 (H,  1, 1.1,  0, 109.47, 3, 120)
 (H,  1, 1.1,  0, 109.47, 4, 120)
 (H,  2, 1.1,  0, 109.47, 1,  80)
 (H,  2, 1.1,  0, 109.47, 6, 120)
 (H,  2, 1.1,  0, 109.47, 7, 120)
 )
 ReturnConnector(
  (
    (0,1,2)
    (0.7, 0,0)
    (0,109.47,0)
    (0,0,120)
    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
  )
 )
 Connector()
 allowed_groups()
)
(
 Z(
 (C, -3, 0.7, -2,   0, -1,   0)
 (N, 0 , 1.2, -3,  179.0, -2,   180)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7, 0,0)
   (0,170,0)
   (0,0,180)
   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
```

```
 )
 Connector()
 allowed_groups()
)
(
 Z(
 (C, -3, 0.7, -2,   0, -1,   0)
 (O,  0, 1.4, -3, 120, -2, -91(180))
 (H,  0, 1.1,  1, 120, -3, 180)
 )
 ReturnConnector(
  (
    (0,1,2)
    (0.7, 0,0)
    (0,109.47,0)
    (0,0,180)
    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
  )
 )
 Connector()
 allowed_groups()
)
(
 Z(
 (O, -3, 1.4, -2,   0,    -1,   0)
 (C,  0, 1.4, -3, 120,    -2, -91(180))
 (H,  1, 1.1,  0, 109.47, -3, 180)
 (H,  1, 1.1,  0, 109.47,  2, 120)
 (H,  1, 1.1,  0, 109.47,  3, 120)
 )
 ReturnConnector(
  (
    (0,1,2)
    (0.7, 0,0)
    (0,109.47,0)
    (0,0,120)
    (0,0,1)
```

```
    (0,0,1)
    (0,0,0)
    ()
  )
 )
 Connector()
 allowed_groups()
)
(
 Z(
 (C, -3, 1.5, -2,   0, -1,   0)
 (O,  0, 1.3, -3, 120, -2, 180)
 (C,  0, 1.5,  1, 120, -3, 180)
 (H,  2, 1.1,  0, 109.47,  1, 180)
 (H,  2, 1.1,  0, 109.47,  3, 120)
 (H,  2, 1.1,  0, 109.47,  4, 120)
 )
 ReturnConnector(
  (
    (0,1,2)
    (0.7, 0,0)
    (0,109.47,0)
    (0,0,180)
    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
  )
 )
 Connector()
 allowed_groups()
)
(
 Z(
 (N, -3, 0.7, -2,   0,    -1,   0)
 (C,  0, 1.4, -3, 120,    -2, -91)
 (O,  1, 1.3,  0, 120,    -3,   0)
 (C,  1, 1.5,  0, 120,     2, 180)
 (H,  3, 1.1,  1, 109.47,  0, 180)
 (H,  3, 1.1,  1, 109.47,  4, 120)
```

```
 (H,  3, 1.1,  1, 109.47,  5, 120)

 (H,  0, 1.1, 1, 120, 2, 180)
 )
 ReturnConnector(
  (
    (0,1,2)
    (0.7, 0,0)
    (0,109.47,0)
    (0,0,10)
    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
  )
 )
 Connector()
 allowed_groups()
)
(
 Z(
 (H,  -3, 0.4, -2, 0, -1, 0)
 )
 ReturnConnector(
  (
    (0,-3,-2)
    (0.4, 0,0)
    (0,109.47,0)
    (0,0,180)
    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
  )
 )
 Connector()
 allowed_groups()
)
(
 Z(
```

```
 (N, -3, 0.7, -2,   0, -1,   0)
 (O,  0, 1.2, -3, 120, -2, -80)
 (O,  0, 1.2,  1, 120, -3, 180)
 )
 ReturnConnector(
  (
    (0,1,2)
    (0.7, 0,0)
    (0,109.47,0)
    (0,0,180)
    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
  )
 )
 Connector()
 allowed_groups()
)
(
 Z(
  (o, -3, 0.6, -2,   0, -1,   0)
  (c,  0, 1.4, -3, 120, -2,   0(180))
  (o,  1, 1.3,  0, 120, -3,   0(180))
  (c,  1, 1.5,  0, 120,  2, 180)
  (H,  3, 1.1,  1, 109.47, 0, 180)
  (h,  3, 1.1,  1, 109.47, 4, 120)
  (h,  3, 1.1,  1, 109.47, 5, 120)
 )
 ReturnConnector(
  (
    (0,1,2)
    (0.7, 0,0)
    (0,120,0)
    (0,0,180)
    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
  )
```

```
 )
 Connector()
 allowed_groups()
)
(
 Z(
 (O, -3, 0.6, -2,   0,    -1,   0)
 (H,  0, 1.1, -3, 120,    -2, -91(180))
 )
 ReturnConnector(
  (
    (0,1,-3)
    (0.7, 0,0)
    (0,109.47,0)
    (0,0,120)
    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
  )
 )
 Connector()
 allowed_groups()
)
(
 Z(
 (N, -3, 0.7, -2,   0   ,    -1,   0)
 (H,  0, 1.1, -3, 120   ,    -2, 180)
 (H,  0, 1.1,  1, 109.47,    -3, 120(-120))
 )
 ReturnConnector(
  (
    (0,1,2)
    (0.7, 0,0)
    (0,109.47,0)
    (0,0,120)
    (0,0,1)
    (0,0,1)
    (0,0,0)
    ()
```

```
   )
  )
  Connector()
  allowed_groups()
 )
 (
 Z()
 ReturnConnector(
  (
   (-3,-2,-1)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   (0,0,0)
   ()
  )
 )
 Connector()
 allowed_groups()
 )

(
 Z(
  (C, -3, 0.7, -2, 0, -1, 0)
  (H,  0, 1.1, -3, 120, -2, 180)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7,0,0)
   (0,120,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  )
```

```
 )
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2,   0, -1,   0)
  (N,  0, 1.4, -3, 120, -2, 180)
  (H,  1, 1.1,  0, 120, -3, -30(150))
  (H,  1, 1.1,  0, 120,  2, 150)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7,0,0)
   (0,120,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  ))
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2,   0, -1,   0)
  (N,  0, 1.4, -3, 120, -2, 180)
  (O,  1, 1.2,  0, 120, -3, 150)
  (O,  1, 1.2,  0, 120,  2, 180)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7,0,0)
   (0,120,0)
   (0,0,180)
```

```
   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  ))
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2,   0, -1,   0)
  (N,  0, 1.4, -3, 120, -2, 180)
  (H,  1, 1.1,  0, 120, -3, -30(150))
  (C,  1, 1.4,  0, 120,  2, 180)
  (O,  3, 1.3,  1, 120,  2, 0(180))
  (C,  3, 1.5,  1, 120,  4, 180)
  (H,  5, 1.1,  3, 109.47, 4,180)
  (H,  5, 1.1,  3, 109.47, 6, 120)
  (H,  5, 1.1,  3, 109.47, 7, 120)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7,0,0)
   (0,120,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  ))
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2,   0,   -1,   0)
  (C,  0, 1.4, -3, 120,   -2, 180)
  (O,  1, 1.3,  0, 120,   -3, -30(150))
```

```
  (C,  1, 1.5,  0, 120,    2, 180)
  (H,  3, 1.1,  1, 109.47, 2, 180)
  (H,  3, 1.1,  1, 109.47, 4, 120)
  (H,  3, 1.1,  1, 109.47, 5, 120)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7,0,0)
   (0,120,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  ))
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2,   0,   -1,   0)
  (O,  0, 1.4, -3, 120,   -2, 180)
  (C,  1, 1.4,  0, 120,   -3, -30(150))
  (O,  2, 1.3,  1, 120,    0, 0(180))
  (C,  2, 1.5,  1, 120,    3, 180)
  (H,  4, 1.1,  2, 109.47, 3,180)
  (H,  4, 1.1,  2, 109.47, 5, 120)
  (H,  4, 1.1,  2, 109.47, 6, 120)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7,0,0)
   (0,120,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
```

```
   (0,0,0)
   ()
  ))
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2, 0, -1, 0)
  (C,  0, 1.5, -3, 120, -2, 180)
  (N,  0, 2.8, 1, 10, -3, 180)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7,0,0)
   (0,120,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  ))
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2,   0, -1,   0)
  (N,  0, 1.4, -3, 120, -2, 180)
  (C,  1, 1.2,  0, 120, -3, 150)
  (C,  1, 1.2,  0, 120,  2, 180)

  (H,  2, 1.1,  1, 109.47, 0,180)
  (H,  2, 1.1,  1, 109.47, 4, 120)
  (H,  2, 1.1,  1, 109.47, 5, 120)

  (H,  3, 1.1,  1, 109.47, 0,180)
  (H,  3, 1.1,  1, 109.47, 7, 120)
```

```
  (H,  3, 1.1,  1, 109.47, 8, 120)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7,0,0)
   (0,120,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  ))
 Connector()
 allowed_groups()
)
(
 Z(
  (C, -3, 0.7, -2,   0, -1,   0)
  (O,  0, 1.4, -3, 120, -2, 180)
  (C,  1, 1.5,  0, 120, -3, -30(150))
  (H,  2, 1.1,  1, 109.47, 0,180)
  (H,  2, 1.1,  1, 109.47, 3, 120)
  (H,  2, 1.1,  1, 109.47, 4, 120)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7,0,0)
   (0,120,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  ))
 Connector()
 allowed_groups()
```

```
)
(
 Z(
  (C, -3, 0.7, -2,   0, -1,   0)
  (S,  0, 1.6, -3, 120, -2, 180)
  (C,  1, 1.5,  0, 120, -3, -30(150))
  (H,  2, 1.1,  1, 109.47, 0,180)
  (H,  2, 1.1,  1, 109.47, 3, 120)
  (H,  2, 1.1,  1, 109.47, 4, 120)
 )
 ReturnConnector(
  (
   (0,1,-3)
   (0.7,0,0)
   (0,120,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
  ))
 Connector()
 allowed_groups()
)
(
 Z(
  (N, -3,0.6, -2,0,-1,0)
 )
 ReturnConnector(
  (
   (0,-3,-2)
   (0.7,0,0)
   (0,120,0)
   (0,0,180)

   (0,0,1)
   (0,0,1)
   (0,0,0)
   ()
```

```
    ))
   Connector()
   allowed_groups()
  )

)
```

%

1 (PDF only) DEFENSE TECHNICAL
INFORMATION CTR
DTIC OCA
8725 JOHN J KINGMAN RD
STE 0944
FORT BELVOIR VA 22060-6218

1 DIRECTOR
US ARMY RESEARCH LAB
IMAL HRA
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RESEARCH LAB
RDRL CIO LL
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RESEARCH LAB
RDRL CIO LT
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
RDRL-WMM-G
B. CHRISTOPHER RINDERSPACHER
JAN W. ANDZELM
TANYA L. CHANTAWANSRI
APG MD 21005

INTENTIONALLY LEFT BLANK.